TECHNISCHE UNIVERSITEIT
EINDHOVEN

FACULTEIT ELEKTROTECHNIEK

vakgroep Informatie- en
Communicatietheorie

De constructie van een energiebegrensde kanaalcode met behulp van enumeratieve technieken.

Door J.J. Wuijts.

Verslag van een stage, verricht in de vakgroep EI, onder leiding van dr.ir. F.M.J. Willems, in de periode juli-september 1991.

Eindhoven, 14 oktober 1991.

Samenvatting.

In deze stage is met behulp van de enumeratieve coderingstechniek een kanaalcode met alfabet $\{0, 1, 2, ..., k\}$ gerealiseerd waarvan de energie begrensd is door een maximale waarde e_{max}.

Voor de enumeratieve coderingsmethode is het nodig om het aantal codewoorden $A(n, e_{max})$ van deze code bij gegeven lengte n en maximale energie e_{max} te bepalen. Er is een algoritme bedacht waarmee deze aantallen $A(n, e_{max})$ berekend kunnen worden.

Er is in Turbo Pascal een unit energie geschreven die de procedures init, encodeer en decodeer bevat. In de procedure init is het algoritme geimplementeerd waarmee de $A(n, e_{max})$ berekend worden. De procedure encodeer codeert een datawoord van de bron naar een energiebegrensd codewoord en de procedure decodeer verricht de omgekeerde taak.

Bovendien is dezelfde energiebegrensde code maar nu met het alfabet $\{1, 3, 5, ..., 2a - 1\}$ gerealiseerd. Dit alfabet komt overeen met de niveau's die gebruikelijk zijn bij Puls Amplitude Modulatie. De gemiddelde energie van de codewoorden die samen een stelsel in de vorm van een n-dimensionale bol vormen blijkt minder te zijn dan bij andere stelsels zoals b.v. de n-dimensionale kubus.

Er is een unit pam geschreven met de procedures init, encodeer en decodeer voor de energiebegrensde code met het alfabet $\{1, 3, 5, ..., 2a - 1\}$.

Er is ook een programma energieminimalisatie geschreven die de winst in gemiddelde energie van de energiebegrensde code met zijn n-dimensionale bolvormige stelsel berekent ten opzichte van de n-dimensionale kubus.

Met het programma simul zijn er simulaties verricht. Van een aantal random gegenereerde codewoorden wordt bekeken of hun gemiddelde energie overeenkomt met de waarde voor de gemiddelde energie zoals die gevonden is met het programma energieminimalisatie.

Tenslotte is met het programma Gaussisch aangetoond dat de fracties van het voorkomen van de niveau's uit $\{1, 3, 5, ..., 2a - 1\}$ gaussisch verdeeld zijn.

Inhoudsopgave

1 Inleiding. 4

2 Een energiebegrensde kanaalcode. 5
- 2.1 De definitie van een energiebegrensde code. 5
- 2.2 De enumeratieve coderingstechniek. 6
- 2.3 Een algoritme voor de berekening van de $B(v,w)$. 10
- 2.4 Een algoritme voor de $A(v,w_{max})$. 12
- 2.5 Het gebruik van de enumeratieve coderingstechniek voor het kanaal. 14

3 De programmatuur voor de energiebegrensde code. 16
- 3.1 De unit energie in Turbo Pascal. 16
  - 3.1.1 Het gebruik van de unit. 16
  - 3.1.2 Beschrijving van de procedure init. 17
  - 3.1.3 Beschrijving van de procedure encodeer. 18
  - 3.1.4 Beschrijving van de procedure decodeer. 18
- 3.2 Het uittesten van de unit energie. 19

4 Energieminimalisatie bij puls amplitude modulatie. 20
- 4.1 N-dimensionale puls amplitude modulatie. 20
- 4.2 Berekening van de gemiddelde energie van een n-dimensionale kubus. . . . 23
- 4.3 Berekening van de gemiddelde energie van een cirkelvormig signaalpuntenstelsel. 25
- 4.4 De gain $G(n)$ voor het n-dimensionale geval. 26
- 4.5 De realisatie van het n-dimensionale bolvormige stelsel met de energiebegrensde code. 27
- 4.6 Berekening van de $A(v,w_{max})$. 29

5 De programmatuur voor n-dimensionale bolvormige puls amplitude modulatie. 31
- 5.1 De unit pam. 31
  - 5.1.1 Het gebruik van de unit. 31
  - 5.1.2 Beschrijving van de procedure init. 32
  - 5.1.3 Beschrijving van de procedure encodeer. 32

5.1.4 Beschrijving van de procedure decodeer. 33
5.2 Het uittesten van de unit pam. 33
5.3 Het programma energieminimalisatie. 34
5.4 Het programma simulatie. 35
5.5 Bespreking van de resultaten van energieminimalisatie en simulatie. 37
5.6 Het programma Gaussisch. 38

6 Conclusies. **42**

Hoofdstuk 1

Inleiding.

Een van de zaken waar de vakgroep Informatie- en Communicatietheorie zich mee bezighoudt is het ontwikkelen van coderingstechnieken. De bekendste coderingstechniek is de enumeratieve coderingsmethode. Mijn opdracht in deze stage is met behulp van deze coderingsmethode een kanaalcode te ontwerpen waarvan de energie begrensd is door een maximale waarde. Bovendien was het mijn taak deze code toe te passen bij Puls Amplitude Modulatie om hiermee een minimalisatie van de gemiddelde uitgezonden energie te realiseren.

Hoofdstuk 2

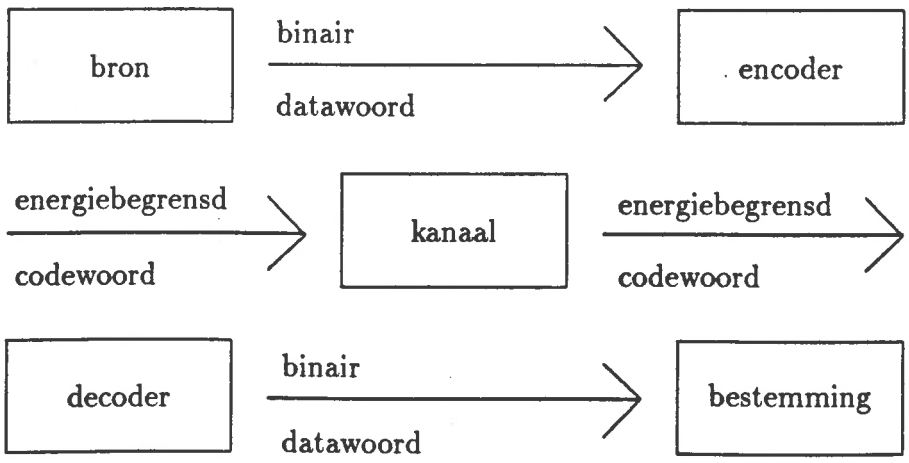
Een energiebegrensde kanaalcode.

2.1 De definitie van een energiebegrensde code.

We stellen elk codewoord van een code voor door een rijtje $x_1, ..., x_n$ met lengte n ,waarbij elk element x_i een waarde kan aannemen uit het alfabet $\{0, 1, 2, 3, ..., k\}$. De energiebegrensde code heeft nu als eigenschap dat voor elk codewoord geldt:

$$\sum_{i=1}^{n} x_i^2 \leq e_{max} \tag{2.1}$$

Hierin is e_{max} dus de maximale energie van alle codewoorden. Het is nu de bedoeling dat data uit een bron wordt gecodeerd naar deze code. Vervolgens worden de codewoorden door een zender over een kanaal heen gezonden. Nadat de codewoorden aangekomen zijn bij de ontvanger worden ze door de decoder weer gedecodeerd naar de oorspronkelijke data van de bron. Dit is schematisch weergegeven in figuur 2.1. Het voordeel van de energiebegrensde



Figuur 2.1: Schema van het encodeer- en decodeerproces.

code is dus dat de zender niet meer energie dan de maximale waarde hoeft te verbruiken.

Nu moeten we een coderingsmethode hebben om de binaire woorden uit de bron te coderen naar de energiebegrensde codewoorden. We kiezen voor de enumeratieve coderingstechniek.

2.2 De enumeratieve coderingstechniek.

De enumeratieve coderingsmethode wordt behandeld in [1, 2, 3, 4, 5]. De theorie wordt zeer helder behandeld in [2] en daarom volgen we in deze paragraaf ook dat artikel.

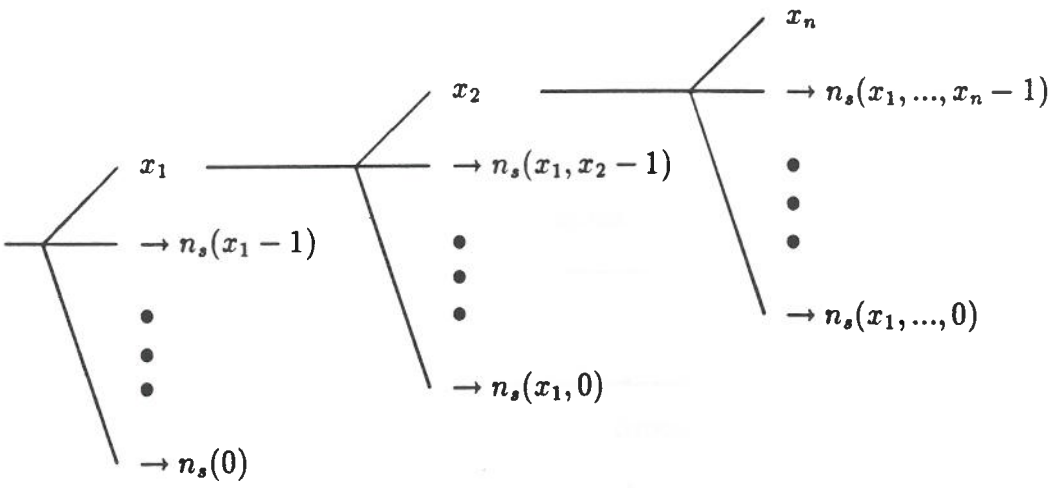
We hebben een set $\{0,1,2,3,...,k\}^n$ van rijtjes ter lengte n. Elk rijtje wordt voorgesteld door een vector $\mathbf{x} = (x_1,...,x_n)$. Elk elementje x_i van de vector $\mathbf{x}$ neemt een waarde aan uit het alfabet $\{0,1,2,3,...,k\}$. We nemen een subset S van $\{0,1,2,3,...,k\}^n$. Het aantal rijtjes van S wordt n_s genoemd en $n_s(x_1,x_2,...,x_j)$ is het aantal rijtjes van S waarvan de eerste j elementen gelijk zijn aan $(x_1,x_2,...,x_j)$. De rijtjes van S worden nu lexicografisch geordend. Onder lexicografische ordening wordt verstaan dat $0 < 1 < 2 < 3 ... < k$. Dus het rijtje $\mathbf{x}$ heeft een kleinere lexicografische index dan $\mathbf{y}$ als $x_j < y_j$ voor de kleinste j waarvoor geldt dat $x_j \neq y_j$.

Dus in het binaire geval bijvoorbeeld $00101 < 00110$. De lexicografische index van $\mathbf{x} \in S \subseteq \{0,1,2,3,...,k\}^n$ wordt nu gegeven door de formule:

$$i_s(\mathbf{x}) = \sum_{j=1}^{n} \sum_{m=0}^{x_j-1} n_s(x_1, x_2, ..., x_{j-1}, m) \tag{2.2}$$

Deze formule kunnen we ook schematisch weergeven met de lexicografische boom in figuur 2.2.

Zoals duidelijk te zien is in figuur 2.2 wordt het aantal rijtjes geteld dat onder het rijtje



Figuur 2.2: De lexicografische boom.

$\mathbf{x} = (x_1,...,x_n)$ ligt. De lexicografische index van $\mathbf{x}$ is dus gelijk aan het aantal rijtjes dat

onder $\mathbf{x}$ ligt. Het laagste rijtje in de boom heeft geen rijtjes meer onder zich en heeft dus index 0. Omdat het aantal rijtjes in S n_s bedraagt moet de index van het hoogste rijtje gelijk zijn aan $n_s - 1$.

In het binaire geval met een subset $S \subseteq \{0,1\}^n$ kan men formule 2.2 schrijven als:

$$i_s(\mathbf{x}) = \sum_{j=1}^{n} x_j n_s(x_1, x_2, ..., x_{j-1}, 0) \tag{2.3}$$

Het terugvinden van een rijtje $\mathbf{x}$ als zijn lexicografische index $i = i_s(\mathbf{x})$ gegeven is gaat volgens het volgende algoritme:

Bepaling van x_1:
- Als $i \geq n_s(0)$ dan verminder i met $n_s(0)$: $i = i - n_s(0)$.
- Als dan voor deze waarde van i geldt: $i \geq n_s(1)$ dan $i = i - n_s(1)$.
- Ga zo door totdat voor een zekere waarde m_1 geldt: $i < n_s(m_1)$.

Dan hebben we voor het eerste element x_1 van $\mathbf{x}$ gevonden:
$x_1 = m_1$.
Bepaling van elk ander element x_j gaat op dezelfde manier:
- Ga uit van de verminderde indexwaarde i zoals die na de bepaling van het element x_{j-1} overblijft.
- Als $i \geq n_s(x_1, x_2, ..., x_{j-1}, 0)$ dan $i = i - n_s(x_1, x_2, ..., x_{j-1}, 0)$.
- Als $i \geq n_s(x_1, x_2, ..., x_{j-1}, 1)$ dan $i = i - n_s(x_1, x_2, ..., x_{j-1}, 1)$.
- Men moet dan zo door gaan totdat men een waarde m_j tegenkomt waarvoor geldt: $i < n_s(x_1, x_2, ..., x_{j-1}, m_j)$. Dan heeft men voor x_j gevonden: $x_j = m_j$.

Nu behandelen we twee voorbeelden in het binaire geval.

Voorbeeld 1: enumeratie van alle binaire n-rijtjes:
$S = \{0,1\}^n$.
Het is duidelijk dat het aantal binaire rijtjes dat begint met een aantal vaste elementen $x_1, x_2, ..., x_{j-1}, 0$ gelijk is aan 2^{n-j}, dus

$$n_s(x_1, x_2, ..., x_{j-1}, 0) = 2^{n-j} \tag{2.4}$$

De lexicografische index volgt met gebruikmaking van formule 2.3:

$$i_s(\mathbf{x}) = \sum_{j=1}^{n} x_j 2^{n-j} \tag{2.5}$$

Voorbeeld 2: enumeratie van rijtjes met gewicht w:
$S = \{\mathbf{x} \in \{0,1\}^n : \sum_{i=1}^{n} x_i = w\}$.

Hier geldt:

$$n_s(x_1, x_2, ..., x_{j-1}, 0) = \begin{pmatrix} n-j \\ n(w,j) \end{pmatrix} \tag{2.6}$$

met:

$$n(w,j) = w - \sum_{i=1}^{j-1} x_i \tag{2.7}$$

Met 2.3 en 2.6 volgt voor de lexicografische index:

$$i_s(\mathbf{x}) = \sum_{j=1}^{n} x_j \begin{pmatrix} n-j \\ n(w,j) \end{pmatrix} \tag{2.8}$$

Na deze twee voorbeelden komt nu als het echte probleem de enumeratie van rijtjes uit de volgende subset:

$S = \{\mathbf{x} \in \{0,1,2,...,k\}^n : \sum_{i=1}^{n} x_i^2 \leq e_{max}\}$.

We definiëren nu $A(n, e_{max})$ als het aantal rijtjes in S die dus de lengte n en maximale energie e_{max} hebben. We hebben al eerder het aantal rijtjes in S n_s genoemd, dus:

$$n_s = A(n, e_{max}) \tag{2.9}$$

We beschouwen nu $n_s(x_1, x_2, ..., x_{j-1}, m)$, het aantal rijtjes die met de elementen $x_1, x_2, ..., x_{j-1}, m$ beginnen. Invullen in formule 2.1 geeft:

$$\sum_{i=1}^{j-1} x_i^2 + m^2 + \sum_{i=j+1}^{n} x_i^2 \leq e_{max} \tag{2.10}$$

Vergelijking 2.10 kan ook zo geschreven worden:

$$\sum_{i=j+1}^{n} x_i^2 \leq e'_{max} \tag{2.11}$$

Hierin is e'_{max} gelijk aan:

$$e'_{max} = e_{max} - \sum_{i=1}^{j-1} x_i^2 - m^2 \tag{2.12}$$

Het is duidelijk dat vergelijking 2.11 de definitie vormt van een andere subset S' met rijtjes ter lengte $n-j$ en maximale energie e'_{max}:
$S' = \left\{\mathbf{x} \in \{0,1,2,...,\mathrm{k}\}^{\mathrm{n-j}} : \sum_{\mathrm{i=1}}^{\mathrm{n-j}} \mathrm{x_i^2} \leq \mathrm{e'_{max}}\right\}$. Uit 2.9 volgt voor het aantal rijtjes in S' $n_{s'}$:

$$n_{s'} = A(n-j, e'_{max}) \tag{2.13}$$

Uit 2.10 en 2.11 blijkt dat:

$$n_s(x_1, x_2, ..., x_{j-1}, m) = n_{s'} \tag{2.14}$$

$$n_s(x_1, x_2, ..., x_{j-1}, m) = A\left(n-j, e_{max} - \sum_{i=1}^{j-1} x_i^2 - m^2\right) \tag{2.15}$$

Met 2.2 hebben we nu voor de lexicografische index:

$$i_s(\mathbf{x}) = \sum_{\mathrm{j=1}}^{\mathrm{n}} \sum_{\mathrm{m=0}}^{\mathrm{x_j-1}} \mathrm{A}\left(\mathrm{n-j}, \mathrm{e_{max}} - \sum_{\mathrm{i=1}}^{\mathrm{j-1}} \mathrm{x_i^2} - \mathrm{m^2}\right) \tag{2.16}$$

Om de lexicografische index te kunnen berekenen met 2.16 moeten we de waarden $A(v, w_{max})$ kunnen berekenen voor $1 \leq v \leq n-1$ en $0 \leq w_{max} \leq e_{max}$. De waarden $A(0, w_{max})$ kunnen we zo beredeneren: volgens 2.15 komt het dus neer op het bepalen van de $n_s(x_1, x_2, ..., x_{n-1}, m)$ voor $0 \leq m \leq x_n - 1$. Het rijtje $\mathbf{x} = (x_1, x_2, ..., x_n) \in S$ heeft per definitie een energie $\leq e_{max}$. Als we het laatste element x_n vervangen door een waarde $m < x_n$ dan heeft het zo verkregen rijtje $(x_1, x_2, ..., m)$ een kleinere energie dan $\mathbf{x}$, dus is de energie $\leq e_{max}$ zodat het rijtje in S ligt. Voor elke m krijgen we telkens één zo'n onder $\mathbf{x}$ liggend rijtje dus:

$$n_s(x_1, x_2, ..., x_{n-1}, m) = A(0, w_{max}) = 1 \tag{2.17}$$

Nu moet er dus een methode gezocht worden om de $A(v, w_{max})$ te berekenen. Hiervoor beschouwen we eerst het eenvoudigere geval:

$S = \{\mathbf{x} \in \{0, 1, 2, ..., k\}^n : \sum_{i=1}^{n} x_i^2 = e\}$

We definiëren $B(n, e)$ als het aantal rijtjes in deze set S met een constante energie e. De waarden voor $A(v, w_{max})$ volgen dan uit de $B(v, w)$:

$$A(v, w_{max}) = \sum_{w=0}^{w_{max}} B(v, w) \tag{2.18}$$

De afleiding van $n_s(x_1, x_2, ..., x_{j-1}, m)$ gaat op dezelfde manier als bij het vorige geval en leidt tot:

$$n_s(x_1, x_2, ..., x_{j-1}, m) = B\left(n - j, e - \sum_{i=1}^{j-1} x_i^2 - m^2\right) \tag{2.19}$$

De lexicografische index wordt hier nu:

$$i_s(\mathbf{x}) = \sum_{j=1}^{n} \sum_{m=0}^{x_j - 1} B\left(n - j, e - \sum_{i=1}^{j-1} x_i^2 - m^2\right) \tag{2.20}$$

Voor het geval $B(0, w)$ geldt:

$$n_s(x_1, x_2, ..., x_{n-1}, m) = B(0, w) = 0 \tag{2.21}$$

Dit is gemakkelijk in te zien: er bestaan geen rijtjes $x_1, x_2, ..., x_{n-1}, m \in S$ onder het rijtje $x_1, ..., x_n$ met energie e want deze rijtjes hebben een lagere energie dan e en dus zitten ze niet in de set S.

Als we een nu een algoritme hebben om de $B(v,w)$ te berekenen ,dan volgt hieruit met 2.18 de $A(v,w_{max})$ en kunnen we dus de lexicografische index bepalen van rijtjes met begrensde energie en rijtjes met constante energie.

2.3 Een algoritme voor de berekening van de $B(v,w)$.

We zoeken nu naar een algoritme om de $B(v,w)$ te berekenen voor $1 \leq v \leq n$ en $0 \leq w \leq e$. Het geval $B(1,w)$ is eenvoudig: we hebben rijtjes $x_1^2 = w$ ter lengte één. Het is duidelijk dat:

$$B(1,w) = \begin{cases} 1 & \text{als } w \in \{0,1,4,...,k^2\} \\ 0 & \text{anders} \end{cases} \tag{2.22}$$

De bepaling van de $B(v,w)$ voor de andere lengtes v gaat als volgt:
Men kan inzien dat er de volgende relatie bestaat tussen de aantallen $B(v,w)$ voor de lengte v en de aantallen $B(v-1,w)$ voor de lengte $v-1$:

$$B(v,w) = \sum_{w_1=0}^{w} B(1,w_1)B(v-1,w-w_1) \tag{2.23}$$

Met gebruikmaking van 2.22 volgt:

$$B(v,w) = \sum_{i=0}^{k} B(v-1,w-i^2) \tag{2.24}$$

Met 2.24 hebben we nu een relatie gevonden waarmee we de rijtjesaantallen voor de lengte v kunnen berekenen uit de aantallen van de vorige lengte $v-1$. Men begint bij $v=2$ en gaat zo door tot $v=n$. Als voorbeeld zijn de met 2.24 verkregen waarden $B(v,w)$ voor $k=2, n=7$ en $e=10$ uitgezet in tabel 2.1. Met behulp van deze tabel en met 2.20 en 2.21 kunnen we de lexicografische index bepalen van elk rijtje uit de set S van rijtjes met alfabet $\{0,1,2\}$,lengte $n=7$ en constante energie $e=10$.
We nemen nu een voorbeeld.
Het rijtje $\mathbf{x} = (2100012)$:

$$i_s(2100012) = B(6,10) + B(6,9) + B(5,6) + B(1,5) + B(0,4) + B(0,3) = 186.$$

In figuur 2.3 staat de lexicografische boom behorende bij dit rijtje.

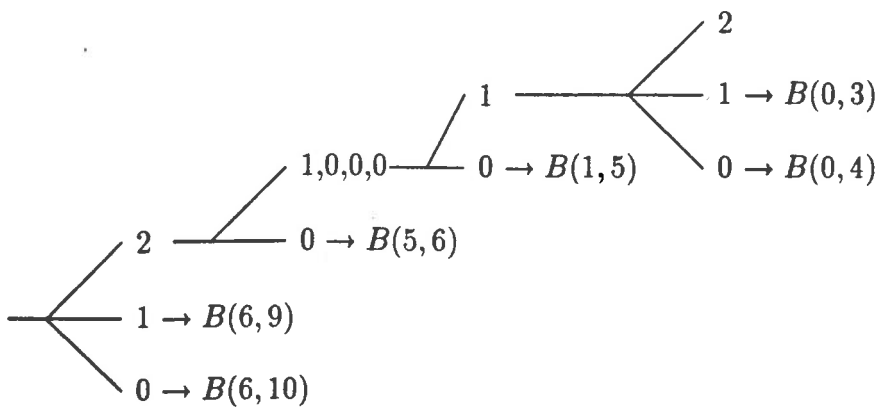
Met het algoritme behandeld op pagina 7 en vergelijking 2.19 kunnen we bij gegeven index het rijtje ook weer terugvinden. We hebben index $i_s = 186$.

bepaling x_1: $i_s = 186 \geq n_s(0) = B(6,10) = 90 \rightarrow i_s = 186 - 90 = 96$.
$i_s = 96 \geq n_s(1) = B(6,9) = 66 \rightarrow i_s = 96 - 66 = 30$.
Dus $x_1 = 2$ want voor de hoogste letter $k=2$ uit het
alfabet geldt altijd $i_s = 30 \leq n_s(2) = B(6,6) = 61$.

De bepaling van x_2 tot en met x_6 gaat op dezelfde manier. Hierna blijft een indexwaarde

w	10	0	0	0	6	30	90	217
	9	0	0	3	12	30	66	147
	8	0	1	3	6	15	45	126
	7	0	0	0	4	20	60	141
	6	0	0	3	12	30	61	112
	5	0	2	6	12	21	36	63
	4	1	2	3	5	10	21	42
	3	0	0	1	4	10	20	35
	2	0	1	3	6	10	15	21
	1	1	2	3	4	5	6	7
	0	1	1	1	1	1	1	1
		1	2	3	4	5	6	7
								v

Tabel 2.1: $B(v,w)$ voor $k=2$, $1 \leq v \leq 7$ en $0 \leq w \leq 10$.



Figuur 2.3: De lexicografische boom van $\mathbf{x} = (2100012)$.

$i_s = 0$ over waaruit x_7 volgt.

$x_7 : i_s = 0$. De i_s moet nu altijd 0 zijn want het laatste element x_n heeft geen onderliggende paden en volgt uit:
$x_n = \sqrt{e - \sum_{i=1}^{n-1} x_i^2}$. Dus $x_7 = 2$.
We hebben nu gevonden $\mathbf{x} = 2100012$, het rijtje waar we mee begonnen zijn.

2.4 Een algoritme voor de $A(v, w_{max})$.

Met formule 2.18 kunnen we nu de $A(v, w_{max})$ afleiden uit de $B(v, w)$:

$$A(v, w_{max}) = \sum_{w=0}^{w_{max}} B(v, w)$$

Met gebruikmaking van 2.24 krijgen we nu:

$$A(v, w_{max}) = \sum_{w=0}^{w_{max}} \sum_{i=0}^{k} B(v-1, w-i^2) \tag{2.25}$$

Dit kunnen we nu als volgt omwerken:

$$A(v, w_{max}) = \sum_{i=0}^{k} \sum_{w=0}^{w_{max}} B(v-1, w-i^2)$$

$$A(v, w_{max}) = \sum_{i=0}^{k} \sum_{w'=0}^{w_{max}-i^2} B(v-1, w')$$

Tenslotte hebben we:

$$A(v, w_{max}) = \sum_{i=0}^{k} A(v-1, w_{max}-i^2) \tag{2.26}$$

Dit is nu dezelfde relatie als 2.24 waarmee de $B(v, w)$ berekend konden worden. Dus de bepaling van de $A(v, w_{max})$ gaat op dezelfde manier: begin bij $v = 2$ en ga door tot $v = n$. De waarden $A(1, w_{max})$ zijn zo eenvoudig te bepalen. Volgens 2.18:

$$A(1, w_{max}) = \sum_{w=0}^{w_{max}} B(1, w) \tag{2.27}$$

Met behulp van 2.22 wordt het nu duidelijk dat $A(1, w_{max})$ gelijk is aan het aantal kwadraten $w \in \{0, 1, 4, ..., k^2\}$ die liggen in het interval $[0, w_{max}]$. Uit 2.27 volgt:

$$A(1, w_{max}) = \begin{cases} A(1, w_{max}-1) + 1 & \text{als } w_{max} \in \{0, 1, 4, ..., k^2\} \\ A(1, w_{max}-1) & \text{anders} \end{cases} \tag{2.28}$$

W_{max} 10	3	9	26	72	182	421	912
9	3	9	26	66	152	331	695
8	3	9	23	54	122	265	548
7	3	8	20	48	107	220	422
6	3	8	20	44	87	160	281
5	3	8	17	32	57	99	169
4	3	6	11	20	36	63	106
3	2	4	8	15	26	42	64
2	2	4	7	11	16	22	29
1	2	3	4	5	6	7	8
0	1	1	1	1	1	1	1
	1	2	3	4	5	6	7
							v

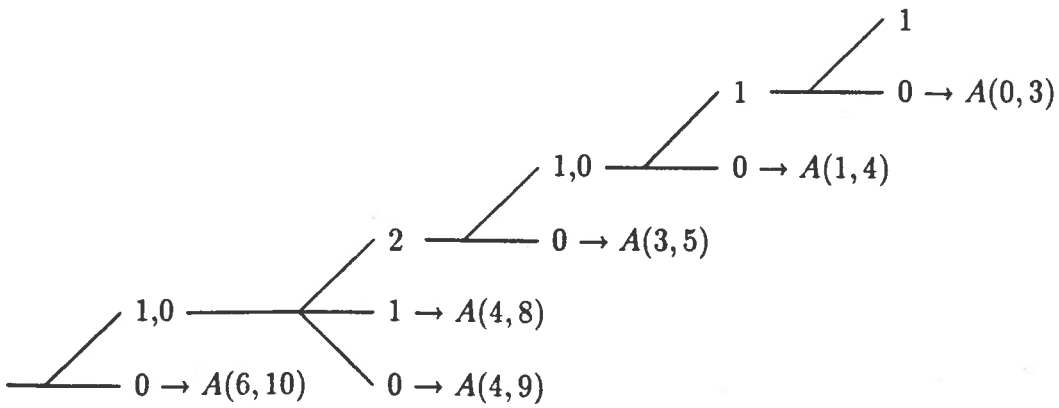
Tabel 2.2: $A(v, w_{max})$ voor $k = 2$, $1 \leq v \leq 7$ en $0 \leq w_{max} \leq 10$.

Men begint bij $A(1,0) = B(1,0)$ $(= 1$ in dit geval) en gaat met 2.28 door tot aan $w_{max} = e_{max}$. Als voorbeeld worden de waarden van $A(v, w_{max})$ bepaald voor het alfabet $\{0,1,2\}, n = 7$ en $e_{max} = 10$. De resultaten staan vermeld in tabel 2.2. Met deze tabel en de vergelijkingen 2.16 en 2.17 kan nu de lexicografische index bepaald worden van elk van de rijtjes met alfabet $\{0,1,2\}, n = 7$ en $e_{max} = 10$. We nemen als voorbeeld het rijtje $\mathbf{x} = (1021011)$. Volgens 2.16:

$$i_s(\mathbf{x}) = A(6,10) + A(4,9) + A(4,8) + A(3,5) + A(1,4) + A(0,3) = 562.$$

In figuur 2.4 staat de lexicografische boom afgebeeld van dit rijtje.

Zo is van een aantal voorbeeldrijtjes de index bepaald en geplaatst in tabel 2.3. Met het



Figuur 2.4: De lexicografische boom van het rijtje $\mathbf{x} = (1021011)$

algoritme behandeld op pagina 7 en vergelijking 2.15 kunnen we het rijtje ook weer uit de index terugvinden. Met $i_s = 562$:

rijtje	index
0000120	15
0000121	16
0000122	17
0000200	18
0000201	19
0000202	20
0000210	21
0000211	22
0000212	23
1021010	561
1021011	562
1021020	563
1021100	564
1021101	565
1021110	566
1021111	567

Tabel 2.3: De index van een aantal voorbeeldrijtjes.

bepaling x_1: $i_s = 562 \geq n_s(0) = A(6,10) = 421 \rightarrow i_s = 562 - 421 = 141$.
$i_s = 141 < n_s(1) = A(6,9) = 331$.
Dus $x_1 = 1$.

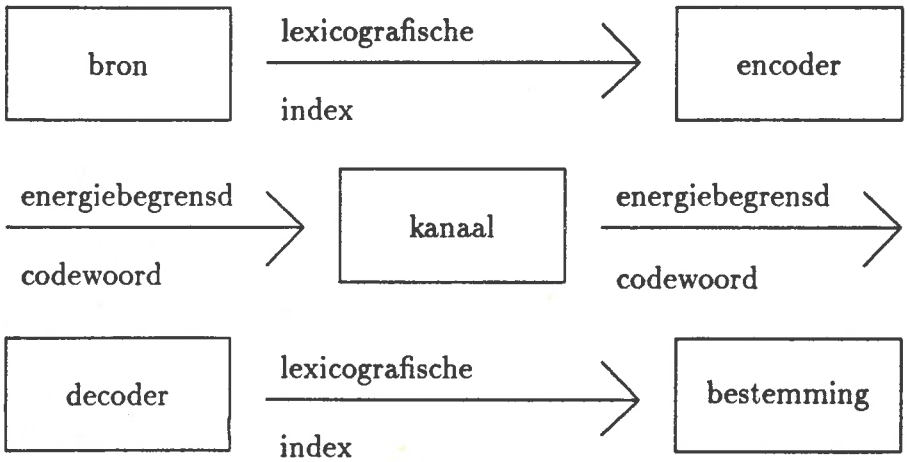
De bepaling van x_2 tot en met x_6 gaat op dezelfde manier.

bepaling x_7: voor de index i_s die na de bepaling van $n-1$
elementen overblijft geldt:
$i_s = \sum_{m=0}^{x_n-1} n_s(x_1,...,x_{n-1},m) = \sum_{m=0}^{x_n-1} 1 = x_n$. Dus: $x_n = i_s$.
Hier: $x_7 = i_s = 1$.
We hebben dus het oorspronkelijke rijtje $\mathbf{x} = (1,0,2,1,0,1,1)$ teruggevonden.

2.5 Het gebruik van de enumeratieve coderingstechniek voor het kanaal.

We zijn nu zover gekomen dat we uit een energiebegrensd rijtje zijn lexicografische index kunnen bepalen en omgekeerd.

We kunnen de enumeratieve coderingstechniek toepassen op het kanaal in figuur 2.1 door de index te gebruiken voor het binaire datawoord. We krijgen nu figuur 2.5 als definitief schema voor het encodeer- en decodeerproces.



Figuur 2.5: Enumeratieve kanaalcodering.

Hoofdstuk 3

De programmatuur voor de energiebegrensde code.

3.1 De unit energie in Turbo Pascal.

3.1.1 Het gebruik van de unit.

Er is in Turbo Pascal de unit energie geschreven die de taken van de encoder en de decoder in figuur 2.5 uitvoert. Deze unit is beschikbaar op floppy in de file energie.pas. De listing is opgenomen in de appendix. De procedure encodeer verzorgt het coderen van de index naar zijn energiebegrensde codewoord en de procedure decodeer decodeert het codewoord weer terug naar de index. Het codewoord $x_1, x_2, ..., x_n$ heeft alfabet $\{0, 1, 2, ..., k\}$ en voor zijn energie geldt vergelijking 2.1.

Voordat men kan gaan encoderen en/of decoderen moet men eerst initialiseren met de procedure init. Men moet via de volgende invoerparameters een aantal gegevens aan deze procedure doorgeven.

-*kin*: de hoogste letter k uit het alfabet $\{0, 1, 2, ..., k\}$ van het codewoord (maximaal 22).
-*nin*: de lengte n van het codewoord $x_1, ..., x_n$ (maximaal 30).
-*aantalindicesin*: Het aantal energiebegrensde codewoorden is gelijk aan $n_s = A(n, e_{max})$. Uiteraard is het aantal mogelijke indexwaarden gelijk aan het aantal codewoorden n_s. Voor de index geldt: $0 \leq i_s \leq n_s - 1$. Uiteraard is het de bedoeling dat het aantal verschillende indexwaarden $A(n, e_{max})$ die de code aankan minstens gelijk moet zijn aan het aantal verschillende indexwaarden die de bron produceert. Men moet het aantal dat de bron levert nu opgeven via *aantalindicesin*. Dan zorgt de procedure init ervoor door e_{max} te kiezen dat $A(n, e_{max}) \geq$ *aantalindicesin*.

Op zijn beurt geeft init via uitvoerparameters de volgende gegevens door aan de gebruiker:

-*emaxuit*: de procedure init kan bepalen hoeveel maximale energie e_{max} er nodig is om ervoor te zorgen dat $A(n, e_{max}) \geq$ *aantalindicesin*. De gebruiker krijgt informatie over de benodigde e_{max} via *emaxuit*.
-*aantalindicesuit*: $A(n, e_{max})$.

Met de procedure encodeer kan men nu telkens een index coderen naar zijn codewoord. Men geeft op als invoerparameter:

-*indexin*: de te coderen index.

Men krijgt terug als uitvoerparameter:

-*s*: Deze array is van het type *symboolrij* = *array*[1..30] of integer. De eerste n elementen $s[1]$ tot en met $s[n]$ bevatten de elementen x_1 tot en met x_n van het codewoord. Men moet in het hoofdprogramma dat gebruik maakt van de unit het array dat men gebruikt voor het codewoord van het type *symboolrij* declareren. Men moet dus b.v. de volgende regel opnemen in het hoofdprogramma: *var codewoord* : *symboolrij*.

Met de procedure decodeer kan men telkens een codewoord weer decoderen naar zijn index. Men geeft op als invoerparameter:

- *s*: het array van het type *symboolrij* waarvan de eerste n elementen het codewoord bevatten.

Men krijgt terug als uitvoerparameter:

- *indexuit*: de gedecodeerde index.

3.1.2 Beschrijving van de procedure init.

De procedure init heeft als taak het berekenen van de $A(v, w_{max})$. De berekende waarden worden opgeslagen in het array $a[x, y]$. In een lus wordt telkens voor een vaste y waarde de $a[x, y]$ berekend beginnend bij $x = 1$ en eindigend bij $x = n$. Er wordt dan gekeken of volgens vergelijking 2.9 deze y voldoende aantal indices $a[n, y]$ oplevert, dus of het aantal minstens gelijk is aan het gevraagde aantal in *aantalindicesin*. Als de $a[n, y]$ niet genoeg is wordt de y met 1 opgehoogd en wordt de lus nog een keer doorlopen. Als de $a[n, y]$ genoeg is wordt de lus gestopt en y bevat de maximale energiewaarde e_{max} die aan de codewoorden wordt toegekend. In de lus wordt nu vergelijking 2.26 toegepast:

Er moet gesommeerd worden over energiewaarden die zowel in de verzameling $\{0, 1, 4, ..., k^2\}$ als in het interval $[0, y]$ moeten zitten. Alle kwadraten $0, 1, ..., k^2$ bevinden zich in het array *kwadraat*. De pointer l wijst naar het grootste element in het array kwadraat dat zich in het interval $[0, y]$ bevindt. Wanneer voor een volgende ronde in de lus de y met 1 is opgehoogd wordt bekeken of het volgende element in het array *kwadraat* , aangewezen door $l+1$ zich in het interval $[0, y+1]$ bevindt. Als dat zo is kan de pointer l met 1 worden opgehoogd, anders blijft de pointer gelijk aan zijn oude waarde. Hierna wijst de pointer l voor $y + 1$ dus weer naar het grootste element uit het array kwadraat dat in het interval $[0, y+1]$ ligt. Van vergelijking 2.27 weten we dat $a[1, y]$ gelijk is aan het aantal kwadraten uit $\{0, 1, 4, ..., k^2\}$ dat ligt in het interval $[0, y]$. Deze kwadraten bevinden zich op de posities 0 tot en met l in het array *kwadraat*.

Dus: $a[1,y] = l+1$. Voor de overige lengtes volgt met formule 2.26:

$$a[x,y] = \sum_{i=0}^{l} a[x-1, y - kwadraat\ [i]\ ] \tag{3.1}$$

3.1.3 Beschrijving van de procedure encodeer.

De procedure encodeer bepaalt het codewoord uit de gegeven index volgens het algoritme behandeld op pagina 13. De eerste $n-1$ elementen $s[lengte]$ worden in een lus bepaald. Volgens vergelijking 2.15:

$$\begin{aligned} &n_s(s[1],...,s[lengte-1],s[lengte]) = a[n-lengte, energie - s[lengte]^2] \\ &energie = y - \textstyle\sum_{i=1}^{lengte-1} s[i]^2 \end{aligned} \tag{3.2}$$

De variabele energie stelt dus de overgebleven maximale energie voor na de bepaling van de elementen $s[1]$ tot en met $s[lengte-1]$. Het element $s[lengte]$ wordt in een lus vanaf 0 telkens met 1 opgehoogd. Dan wordt de variabele $verminderdeindex$ berekend:

$$\begin{aligned} &verminderdeindex = index - n_s(s[1],...,s[lengte-1],s[lengte]) = \\ &index - a[n-lengte, energie - s[lengte]^2] \end{aligned} \tag{3.3}$$

Als geldt: $verminderdeindex \geq 0$ dan wordt de waarde van $index$:

$$index = verminderdeindex = index - n_s(s[1],...s[lengte-1],s[lengte]) \tag{3.4}$$

De index wordt dus verlaagd volgens het algoritme op pagina 13 en daarna wordt de lus opnieuw doorlopen. De lus moet gestopt worden wanneer $index - n_s(s[1],...,s[lengte-1],s[lengte]) < 0$, dus wanneer $verminderdeindex < 0$ of wanneer $s[lengte] = k$ (want dan is $verminderdeindex$ altijd < 0). Zo bevat $s[lengte]$ na het verlaten van de lus de juiste gezochte waarde. Het laatste element wordt:

$$s[n] = index \tag{3.5}$$

3.1.4 Beschrijving van de procedure decodeer.

Volgens formule 2.16 levert elk element $s[lengte]$ een bijdrage aan de lexicografische index ter grootte:

$$\begin{aligned} &\textstyle\sum_{i=0}^{s[lengte]-1} a[n-lengte, energie - i^2] \\ &energie = y - \textstyle\sum_{j=1}^{lengte-1} s[j]^2 \end{aligned} \tag{3.6}$$

Voor de elementen $s[1]$ tot en met $s[n-1]$ wordt in een lus telkens deze bijdrage uitgerekend en bij de variabele index opgeteld. Voor het laatste element $s[n]$ wordt de bijdrage:

$$\sum_{i=0}^{s[n]-1} a[0, energie - i^2] = \sum_{i=0}^{s[n]-1} 1 = s[n]. \tag{3.7}$$

Met het toevoegen van deze laatste bijdrage aan de variabele index is deze nu gelijk geworden aan de lexicografische index van het codewoord $s[1], ..., s[n]$.

3.2 Het uittesten van de unit energie.

De werking van de procedure init is apart gecontroleerd door aan het einde van deze procedure een schrijfopdracht te plaatsen die de getallen $a[x, y]$ op het scherm plaatst. Deze getallen bleken te kloppen met de getallen $a[x, y]$ in tabel 2.2: de procedure init werkt dus goed. Hierna zijn de schrijfopdrachten weer uit init verwijderd.

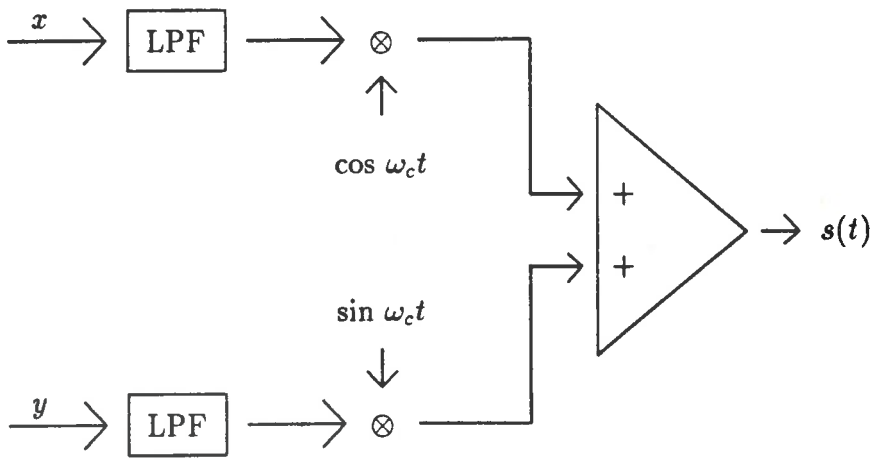
Vervolgens is er een testprogramma geschreven om de procedures encodeer en decodeer uit te testen. Het programma heet program test, staat op floppy in de file entest.pas en de listing ervan is opgenomen in de appendix. Het programma vraagt om een index en schrijft deze ingevoerde index op het scherm. Daarna wordt het codewoord verkregen met encodeer naar het scherm geschreven. Tenslotte wordt de index weer terug bepaald met decodeer en deze laat het programma ook zien op het scherm. De uitvoer van het programma test is nu omgeleid naar de file enuit (op floppy en in de appendix). Aan het programma zijn de indices van de voorbeeldrijtjes uit tabel 2.3 opgegeven. De index gedecodeerd met decodeer blijkt gelijk te zijn aan de oorspronkelijk ingevoerde index en de codewoorden geproduceerd door encodeer komen overeen met die in de tabel zodat geconcludeerd mag worden dat de unit energie goed werkt.

Hoofdstuk 4

Energieminimalisatie bij puls amplitude modulatie.

4.1 N-dimensionale puls amplitude modulatie.

In dit hoofdstuk volgen we nu de stof zoals behandeld in [6]. In figuur 4.1 wordt een Quadrature Amplitude Modulatie modulator schematisch weergegeven. Voor het signaal $s(t)$



Figuur 4.1: QAM modulator.

dat uitgezonden wordt geldt:

$$S(t) = x \cos \omega_c t + y \sin \omega_c t \tag{4.1}$$

De functies $\varphi_1 = \cos \omega_c t$ en $\varphi_2 = \sin \omega_c t$ staan loodrecht op elkaar volgens de definitie van inprodukt in [7]:

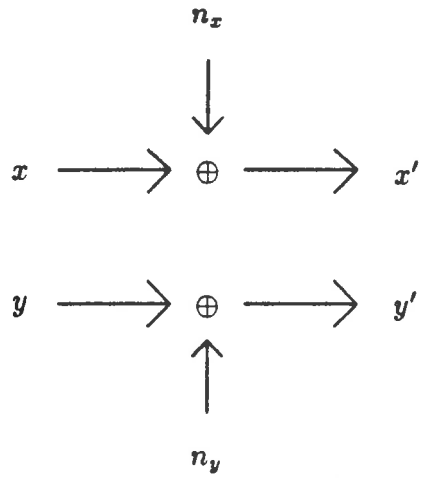
$$(\varphi_1, \varphi_2) = \int_0^{T_c} \varphi_1 \varphi_2 \, dt = \int_0^{T_c} \cos \omega_c t \sin \omega_c t \, dt = 0 \tag{4.2}$$

Bovendien geldt voor de lengte van de functies:

$$\frac{|\varphi_1|^2}{|\varphi_2|^2} = \frac{(\varphi_1, \varphi_1)}{(\varphi_2, \varphi_2)} = \frac{\int_0^{T_c} \cos^2 \omega_c t \, dt}{\int_0^{T_c} \sin^2 \omega_c t \, dt} = 1 \tag{4.3}$$

We kunnen de zaak dus zo voorstellen als in figuur 4.2:

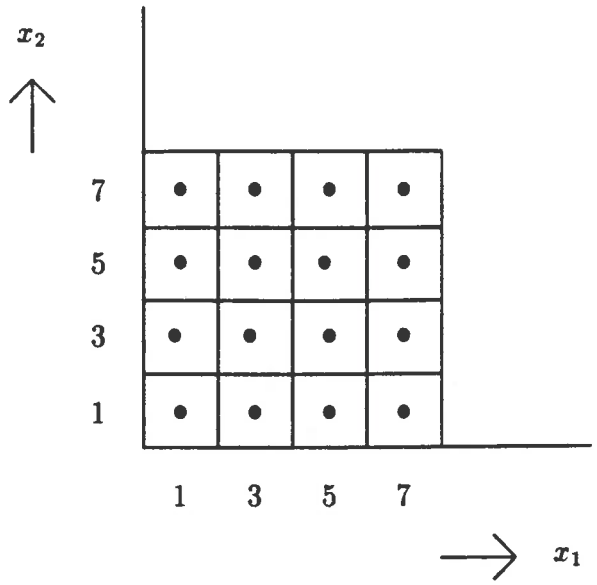
Er worden dus twee-dimensionale vectoren (x, y) over een twee-dimensionaal kanaal ge-



Figuur 4.2: kanaalmodel voor QAM.

zonden. Een paar (x, y) noemen we een signaalpunt of een symbool. De twee coördinaten x en y worden onafhankelijk van elkaar verzonden over aparte parallele kanalen en gestoord door Gaussische ruis variabelen n_x en n_y met gemiddelde 0 en variantie N. Aan de ontvangstzijde worden dan de signaalcoördinaten $x' = x + n_x$ en $y' = y + n_y$ ontvangen. In narrow-sense QAM wordt voor elke signaalcoördinaat Pulse Amplitude Modulatie toegepast. Dat betekent per definitie dat elke coördinaat één van de niveau's uit de verzameling $\{-(2a-1), ..., -5, -3, -1, 1, 3, 5, ..., 2a-1\}$ kan aannemen. In deze stage maken we om het eenvoudig te houden alleen gebruik van de positieve niveau's: $\{1, 3, 5, ..., 2a-1\}$. Het getal a staat hier nu voor het aantal niveau's. Men kan nagaan dat voor alleen de positieve niveau's dezelfde uitspraken gelden als voor de positieve en negatieve niveau's allebei waar artikel [6] mee verder gaat. In het algemeen kan men nu n-dimensionale Puls Amplitude Modulatie toepassen door n-dimensionale vectoren $\mathbf{x} = (x_1, ..., x_n)$ over een kanaal heen te zenden waarbij elk element x_i volgens de 1-dimensionale PAM één van de niveau's $\{1, 3, 5, ..., 2a-1\}$ kan aannemen. De vectoren $\mathbf{x}$ worden gestoord door n-dimensionale ruisvectoren $\mathbf{n} = (n_1, ..., n_n)$.

We tekenen nu voor het twee-dimensionale geval in figuur 4.3 de signaalpunten $\mathbf{x} = (x_1, x_2)$ in het vectorvlak. Het is duidelijk dat als elke coördinaat een waarde uit $\{1, 3, 5, ..., 2a-1\}$ kan aannemen de signaalpunten in een vierkant liggen. We beschouwen nu het encodeer en decodeer schema in figuur 4.4. De encoder bepaalt een signaalvector $\mathbf{x}$ bij het datawoord d ter lengte b bits dat door de bron wordt geleverd. Terwijl de vector $\mathbf{x}$ over het kanaal heengaat wordt hij gestoord door de Gaussische ruisvector $\mathbf{n}$, zodat



Figuur 4.3: De signaalpunten in het vectorvlak voor $a = 4$.

de gestoorde vector $\mathbf{x}' = \mathbf{x} + \mathbf{n}$ bij de ontvangstzijde aankomt. De decoder bepaalt het datawoord d' dat bij de vector $\mathbf{x}'$ hoort. Door transmissiefouten kan $d' \neq d$ zijn.

Als de bron binaire datawoorden levert met een lengte van b bits dan is het aantal verschillende datawoorden dus 2^b. Het aantal verschillende signaalvectoren is uiteraard gelijk aan het aantal verschillende datawoorden 2^b.

Voor het aantal vectoren in het vierkantvormige stelsel van figuur 4.3 geldt:

$$2^b = a^2. \tag{4.4}$$

Voor het aantal niveau's a kan men dus schrijven:

$$a = 2^{b/2} \tag{4.5}$$

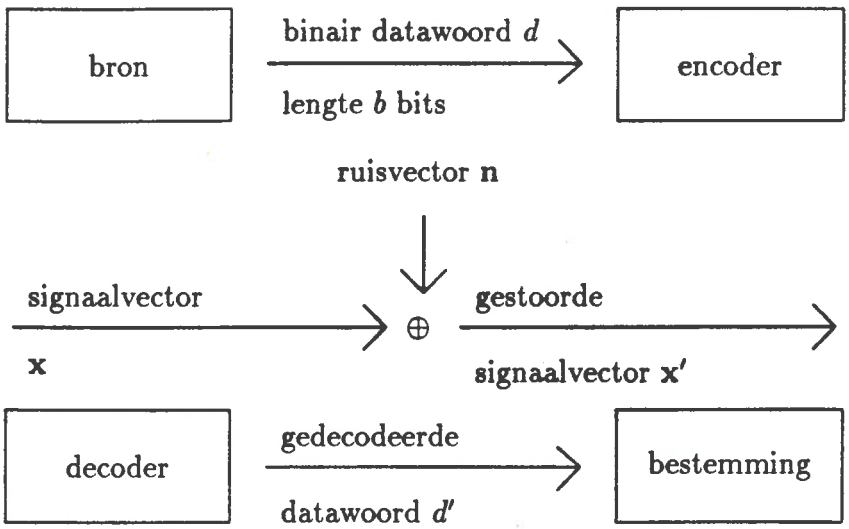
We definiëren nu het aantal bits per dimensie d:

$$d = b/2 \tag{4.6}$$

zodat geldt:

$$a = 2^d \tag{4.7}$$

Uit formule 4.6 volgt de voorwaarde voor een vierkantvormig stelsel dat b even moet zijn. Als niet aan deze voorwaarde voldaan is kunnen de signaalpunten samen geen vierkant vormen. In figuur 4.3 is te zien dat elk signaalpunt een vierkantje om zich heeft met oppervlakte $A = 4$. Dus voor de oppervlakte B van het vierkant dat alle signaalpunten omsluit geldt:



Figuur 4.4: Het encodeer en decodeer schema voor n-dimensionale PAM.

$$B = A2^b = 42^b \tag{4.8}$$

We kunnen figuur 4.3 uitbreiden naar het algemene n-dimensionale geval: een stelsel signaalpunten in de vorm van een n-dimensionale kubus. In het algemene geval wordt vergelijking 4.4:

$$2^b = a^n \tag{4.9}$$

Voor a kan geschreven worden:

$$a = 2^{b/n} = 2^d \tag{4.10}$$

De algemene definitie voor het aantal bits per dimensie d luidt:

$$d = b/n \tag{4.11}$$

Uit formule 4.11 volgt in het algemeen de voorwaarde dat b deelbaar door n moet zijn willen de signaalpunten een n-dimensionale kubus kunnen vormen.

4.2 Berekening van de gemiddelde energie van een n-dimensionale kubus.

De energie van 1-dimensionale PAM gemiddeld over de niveau's $\{1, 3, 5, ..., 2a-1\}$ is per definitie gelijk aan:

$$S_{1,PAM} = \frac{\sum_{t=1}^{a}(2t-1)^2}{a} = \frac{4a^2-1}{3} \tag{4.12}$$

Met vergelijking 4.10 kan men ook schrijven:

$$S_{1,PAM} = \frac{42^{\frac{2b}{n}} - 1}{3} = \frac{42^{2d} - 1}{3} \tag{4.13}$$

De gemiddelde energie van de signaalvectoren $\mathbf{x}$ in een stelsel met de vorm van een n-dimensionale kubus is gelijk aan:

$$S_{n,kubus} = \overline{|\mathbf{x}|^2} = \overline{x_1^2 + ... + x_n^2} = \frac{\sum_{x_1} \cdots \sum_{x_n} x_1^2 + ... + x_n^2}{a^n} = nS_{1,PAM} \tag{4.14}$$

Met de vergelijkingen 4.12 en 4.13:

$$S_{n,kubus} = n\frac{4a^2 - 1}{3} = n\frac{42^{\frac{2b}{n}} - 1}{3} = n\frac{42^{2d} - 1}{3} \tag{4.15}$$

Wanneer het aantal signaalvectoren groot is, dus ook a, kan men $S_{n,kubus}$ in v.g.l. 4.15 benaderen door $\hat{S}_{n,kubus}$:

$$\hat{S}_{n,kubus} = n\frac{4a^2}{3} = n\frac{42^{\frac{2b}{n}}}{3} = n\frac{42^{2d}}{3} \tag{4.16}$$

In het geval van een vierkantvormig stelsel worden de v.g.l. 4.15 en 4.16:

$$S_{2,vierkant} = 2\frac{4a^2 - 1}{3} = 2\frac{42^b - 1}{3} = 2\frac{42^{2d} - 1}{3} \tag{4.17}$$

$$\hat{S}_{2,vierkant} = 2\frac{4a^2}{3} = 2\frac{42^b}{3} = 2\frac{42^{2d}}{3} \tag{4.18}$$

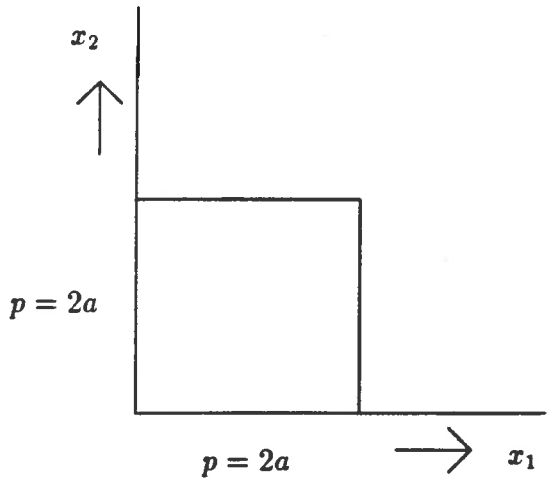
De benadering voor de gemiddelde energie $\hat{S}_{n,kubus}$ kan men ook vinden door de energie gemiddeld over alle punten van de n-dimensionale kubus, dus niet alleen de signaalpunten, te berekenen. In figuur 4.3 kan men duidelijk zien dat de lengte van elke zijde p van het vierkant gelijk is aan $2a$. Het vierkant is nog eens weergegeven in figuur 4.5. Voor de gemiddelde energie van een vierkant geldt:

$$\hat{S}_{2,vierkant} = \overline{|\mathbf{x}|^2} = \frac{\int_0^p \int_0^p x_1^2 + x_2^2 dx_1 dx_2}{p^2} \tag{4.19}$$

In het algemeen wordt de gemiddelde energie van een n-dimensionale kubus dus gegeven door:

$$\hat{S}_{n,kubus} = \overline{|\mathbf{x}|^2} = \frac{\int_0^p \cdots \int_0^p x_1^2 + \cdots + x_n^2 \, dx_1 \cdots dx_n}{p^n} = n\frac{4a^2}{3} \tag{4.20}$$

Dit is inderdaad hetzelfde als datgene wat we met formule 4.16 gevonden hebben.



Figuur 4.5: Berekening van de gemiddelde energie van een vierkant.

4.3 Berekening van de gemiddelde energie van een cirkelvormig signaalpuntenstelsel.

We zouden nu kunnen gaan zoeken naar andere configuraties die een kleinere gemiddelde energie hebben dan het vierkant. De configuratie met de kleinste gemiddelde energie is de cirkel: de geometrische figuur met de minste gemiddelde energie voor een gegeven oppervlakte. In het n-dimensionale geval is dat de n-dimensionale bol. In deze stage hebben we slechts een kwart van de cirkel omdat we alleen gebruik maken van de positieve niveau's $\{1, 3, 5, ..., 2a-1\}$. Al eerder is aangeduid dat beperking tot positieve niveau's voor de theoretische beschouwingen niets uitmaakt. In het volgende wordt het kwart gedeelte van de cirkel gewoon genoemd: de cirkel.

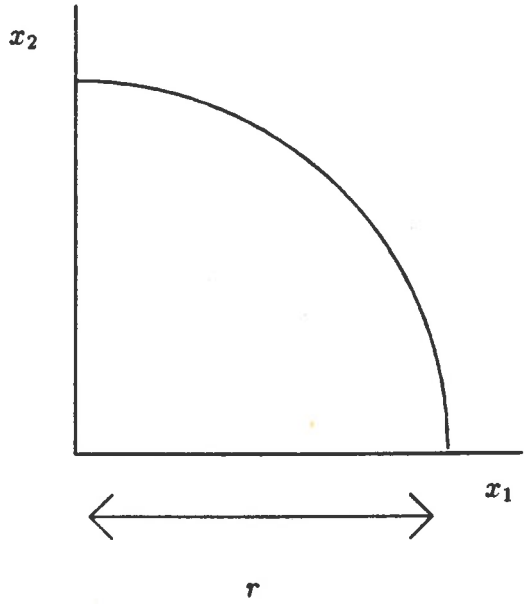
Het cirkelvormige signaalstelsel wordt nu afgebeeld in figuur 4.6. De gemiddelde energie $\hat{S}_{2,cirkel}$ van de cirkelvormige configuratie met straal R in figuur 4.6 is gelijk aan:

$$\hat{S}_{2,cirkel} = \frac{\int_0^{\pi/2} \int_0^R r^2 r dr d\varphi}{\frac{\pi R^2}{4}} = \frac{R^2}{2} \tag{4.21}$$

We definiëren nu de gain $G(2)$ als de winst in de gemiddelde energie die we met het cirkelvormige stelsel kunnen behalen ten opzichte van het vierkantvormige stelsel:

$$G(2) = \frac{\hat{S}_{2,vierkant}}{\hat{S}_{2,cirkel}} \tag{4.22}$$

We vergelijken nu de gemiddelde energie van het cirkelvormige stelsel in figuur 4.6 met de gemiddelde energie van het vierkant in figuur 4.5 waarbij we de oppervlakten van beiden aan elkaar gelijk moeten nemen:



Figuur 4.6: Berekening van de gemiddelde energie van een cirkelvormige configuratie.

$$\frac{\pi R^2}{4} = (2a)^2 \tag{4.23}$$

Hieruit volgt:

$$\frac{R^2}{2} = \frac{8a^2}{\pi} \tag{4.24}$$

De gemiddelde energie van de cirkel kunnen we dus uitdrukken in a:

$$\hat{S}_{2,cirkel} = \frac{8a^2}{\pi} \tag{4.25}$$

De gain $G(2)$ volgt nu met gebruikmaking van vergelijking 4.18:

$$G(2) = \frac{\pi}{3} \tag{4.26}$$

We kunnen de waarde van $G(2)$ ook weergeven in decibels:

$$G(2) = 10 \log \left(\frac{\pi}{3}\right) = 0,20\ dB \tag{4.27}$$

4.4 De gain $G(n)$ voor het n-dimensionale geval.

In artikel [6] wordt een algemene formule gegeven voor de gain $G(n)$ voor een n-dimensionaal bolvormig stelsel ten opzichte van een n-dimensionaal kubusvormig stelsel:

dimensie n	gain $G(n)$	$G(n)$ in dB
2	1,05	0,20
4	1,11	0,45
8	1,18	0,73
16	1,25	0,98
24	1,29	1,10
32	1,31	1,17
48	1,34	1,26
64	1,35	1,31

Tabel 4.1: De gain voor een aantal dimensies.

$$G(n) = \frac{\pi(n+2)}{12}\left(\left(\frac{n}{2}\right)!\right)^{\frac{-2}{n}} \text{ voor even n} \tag{4.28}$$

Voor $n = 2$ vinden we met 4.28: $G(2) = \pi/3$ hetgeen we in 4.26 ook gevonden hadden.

Voor een aantal waarden van n is $G(n)$ berekend met 4.28. De uitkomsten worden nu vermeld in tabel 4.1. We kunnen nog voor het limietgeval dat n naar ∞ gaat een waarde voor de gain afleiden. Volgens de benadering van Stirling geldt:

$$(x!)^{\frac{-1}{x}} \rightarrow \frac{e}{x} \text{ voor } x \rightarrow \infty \tag{4.29}$$

Toepassing van v.g.l. 4.29 op 4.28 levert:

$$G(\infty) = \lim_{n\to\infty} G(n) = \frac{\pi e}{6} \tag{4.30}$$

Of in decibel:

$$G(\infty) = 1,53\ dB \tag{4.31}$$

4.5 De realisatie van het n-dimensionale bolvormige stelsel met de energiebegrensde code.

We hebben de energiebegrensde code gedefinieerd als een code die bestaat uit codewoorden $x_1, ..., x_n$ waarvan elk element x_i een waarde kan aannemen uit het alfabet $\{0, 1, 2, 3, ..., k\}$ en waarvoor v.g.l. 2.1 geldt.

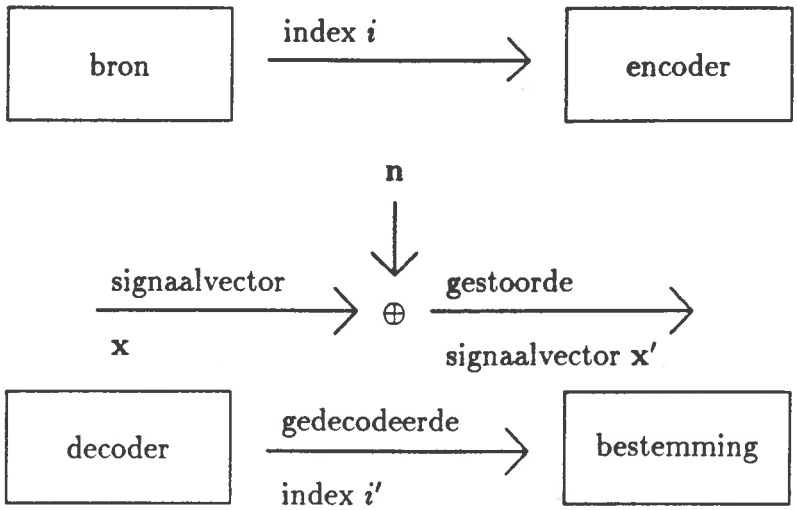
Het is duidelijk dat de codewoorden $x_1, ..., x_n$ gebruikt kunnen worden als de signaalvectoren $\mathbf{x} = (x_1, ..., x_n)$ bij n-dimensionale PAM. Voor de lengte van de vectoren geldt dan volgens 2.1:

$$|\,\mathbf{x}\,| \leq \sqrt{e_{max}} \tag{4.32}$$

De signaalvectoren $\mathbf{x}$ liggen dus in een n-dimensionale bol met straal R:

$$R = \sqrt{e_{max}} \tag{4.33}$$

We kunnen dus met de energiebegrensde code een n-dimensionaal bolvormig stelsel van signaalvectoren realiseren. Het encodeer- en decodeer schema van figuur 4.4 komt er dus voor het bolvormige geval uit te zien als weergegeven in figuur 4.7. Nu moeten we voor



Figuur 4.7: N-dimensionale bolvormige PAM m.b.v. de energiebegrensde code.

de energiebegrensde code nu nog wel een ander alfabet $\{1, 3, 5, ..., 2a-1\}$ bestaande uit de PAM niveau's nemen. De vergelijkingen 2.15, 2.16 en 2.17 blijven uiteraard geldig. In 2.15 en 2.16 wordt een kleine aanpassing aangebracht voor de niveau's $2m-1$ met $1 \leq m \leq a$:

$$n_s(x_1, ..., x_{j-1}, 2m-1) = A\left(n-j, e_{max} - \sum_{i=1}^{j-1} x_i^2 - (2m-1)^2\right) \tag{4.34}$$

$$i_s(\mathbf{x}) = \sum_{j=1}^{n} \sum_{m=1}^{\frac{x_j-1}{2}} A\left(n-j, e_{max} - \sum_{i=1}^{j-1} x_i^2 - (2m-1)^2\right) \tag{4.35}$$

Met gebruikmaking van formule 2.17 kunnen we voor 4.35 nog schrijven:

$$i_s(\mathbf{x}) = \sum_{j=1}^{n-1} \sum_{m=1}^{\frac{x_j-1}{2}} A\left(n-j, e_{max} - \sum_{i=1}^{j-1} x_i^2 - (2m-1)^2\right) + \frac{x_n - 1}{2} \tag{4.36}$$

Nu moeten we voordat we $i_s(\mathbf{x})$ met 4.36 kunnen bepalen de getallen $A(v, w_{max})$ berekenen.

W_{max} 18	2	4	4	5	W_{max} 37	3	8	17	32
17	2	3	4	5	36	3	8	17	32
16	2	3	4	5	35	3	8	17	19
15	2	3	4	5	34	3	8	11	19
14	2	3	4	5	33	3	6	11	19
13	2	3	4	5	32	3	6	11	19
12	2	3	4	5	31	3	6	11	19
11	2	3	4	1	30	3	6	11	19
10	2	3	1	1	29	3	6	11	19
9	2	1	1	1	28	3	6	11	19
8	1	1	1	1	27	3	6	11	11
7	1	1	1	1	26	3	6	7	11
6	1	1	1	1	25	3	4	7	11
5	1	1	1	1	24	2	4	7	11
4	1	1	1	1	23	2	4	7	11
3	1	1	1	0	22	2	4	7	11
2	1	1	0	0	21	2	4	7	11
1	1	0	0	0	20	2	4	7	11
0	0	0	0	0	19	2	4	7	5
	1	2	3	4		1	2	3	4
									v

Tabel 4.2: $A(v, w_{max})$ voor het alfabet $(1,3,5)$, $n = 4$ en $e_{max} = 36$.

4.6 Berekening van de $A(v, w_{max})$.

De vergelijkingen 2.26 en 2.28 worden nu met het nieuwe alfabet:

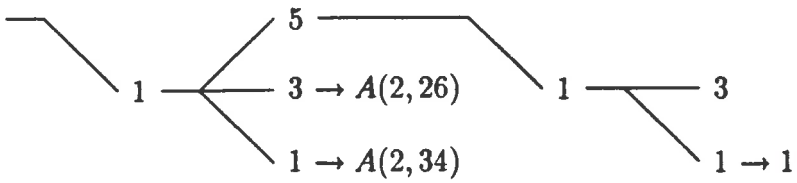
$$A(v, w_{max}) = \sum_{i=1}^{a} A(v-1, w_{max} - (2i-1)^2) \tag{4.37}$$

$$A(1, w_{max}) = \begin{cases} A(1, w_{max}-1) + 1 & \text{als } w_{max} \in \{1, 9, 25, ..., (2a-1)^2\} \\ A(1, w_{max}-1) & \text{anders} \end{cases} \tag{4.38}$$

Als voorbeeld zijn de $A(v, w_{max})$ met 4.37 en 4.38 berekend voor het alfabet $\{1,3,5\}, n = 4$, en $e_{max} = 36$. De resultaten staan vermeld in tabel 4.2. We bepalen nu als voorbeeld de lexicografische index $i_s(\mathbf{x})$ van de vector $\mathbf{x} = (1,5,1,3)$ met 4.36:

$$i_s(\mathbf{x}) = A(2,34) + A(2,26) + 1 = 15.$$

In figuur 4.8 staat de lexicografische boom van dit rijtje. De terugbepaling van het rijtje $(1,5,1,3)$ uit de index gelijk aan 15 gaat volgens het bekende algoritme, gebruikmakend van 4.34.



Figuur 4.8: Lexicografische boom van het rijtje $\mathbf{x} = (1, 5, 1, 3)$.

rijtje	index
1111	0
1113	1
1115	2
1151	6
1153	7
1311	8
1313	9
1315	10
1331	11
1333	12
1351	13
1511	14
1513	15
5311	31

Tabel 4.3: De index van een aantal voorbeeldrijtjes.

Bepaling x_1: $i_s = 15 < n_s(2 * 1 - 1) = A(3, 35) = 17$.
Dus $x_1 = 1$.
Bepaling van x_2 tot en met x_3 gaat op dezelfde manier.
Bepaling x_4: Uit de laatste term van 4.36
volgt dat er na bepaling van $n - 1$ elementen een
index i_s overblijft ter grootte:
$i_s(\mathbf{x}) = \frac{x_n - 1}{2}$
Dus x_n wordt: $x_n = 2i_s(\mathbf{x}) + 1$. Hier $x_4 = 3$.

In tabel 4.3 staan een aantal voorbeeldrijtjes met hun index.

Hoofdstuk 5

De programmatuur voor n-dimensionale bolvormige puls amplitude modulatie.

5.1 De unit pam.

5.1.1 Het gebruik van de unit.

Er is een unit pam geschreven die het encodeer en decodeerproces van figuur 4.7 uitvoert. Deze unit staat in de file pam.pas en zijn listing is opgenomen in de appendix. Deze unit werkt hetzelfde als de unit energie, maar nu voor het alfabet $\{1, 3, 5, ..., 2a-1\}$.

Aan de procedure init die voor het coderen en/of decoderen aangeroepen moet worden moeten hier de volgende invoerparameters opgegeven worden:

-*aantallevelsin*: het aantal niveau's a. Het getal a was ook alweer gelijk aan het aantal niveau's in het alfabet $\{1, 3, 5, ..., 2a-1\}$. De maximale waarde die men voor a kan invoeren bedraagt 11.
-*nin*: de te gebruiken dimensie. De maximale toegestane waarde is 30.
- *aantalindicesin* : hier moet de gebruiker weer het aantal verschillende indexwaarden die de bron levert via deze invoerparameter opgeven.

Men krijgt van init terug als uitvoerparameter:

- *aantalindicesuit*: Het aantal verschillende indexwaarden *aantalindicesuit* die de code aankan wordt minstens gelijk gemaakt aan het aantal dat de bron levert:
$aantalindicesuit = A(n, e_{max}) \geq aantalindicesin.$

-*emaxuit*: De maximale energie e_{max} die hiervoor nodig is.

De procedure encodeer codeert de index naar de signaalvector. Invoerparameter:

- *indexin*: de te coderen index.

Uitvoerparameter:

- s: het array van het type *symboolrij* waarvan de eerste n elementen de signaalvector bevatten. De gebruiker moet ook hier niet vergeten om het array in zijn hoofdprogramma dat hij voor de signaalvector gebruikt van het type *symboolrij* te declareren.

De procedure decodeer bepaalt de index weer terug uit de signaalvector.

Invoerparameter:

-s: het array met de signaalvector die men aan decodeer moet opgeven.

Uitvoerparameter:

- *indexuit* : de index die decodeer bepaalt heeft uit de signaalvector.

5.1.2 Beschrijving van de procedure init.

De werking van de procedures in de unit pam is identiek aan de werking van de procedures in de unit energie. Er zijn een aantal aanpassingen aangebracht voor het alfabet $\{1, 3, ..., 2a-1\}$. Deze aanpassingen worden nu kort toegelicht.

-Het array *kwadraat* bevat nu de kwadratische energiewaarden
$1, 9, 25, ..., (2 * aantallevels - 1)$ op de posities 1 tot en met *aantallevels*.

-Het gedeelte van de kwadraten uit het array *kwadraat* dat zich voor een bepaalde y-waarde in de lus in het interval $[0, y]$ bevindt loopt nu van positie 1 tot en met l.

-Volgens 4.38 is $a[1, y]$ gelijk aan het aantal kwadraten uit
$\{1, 9, 25, ..., (2 * aantallevels - 1)\}$ dat in het interval $[0, y]$ ligt. Daar deze kwadraten in het array *kwadraat* op de posities 1 tot en met l zitten is het duidelijk dat: $a[1, y] = l$.

-De overige waarden $a[x, y]$ voor $2 \leq x \leq n$ worden volgens 4.37 berekend met:

$$a[x, y] = \sum_{i=1}^{l} a[x-1, y - kwadraat\,[i]\,] \tag{5.1}$$

5.1.3 Beschrijving van de procedure encodeer.

De bepaling van de eerste $n-1$ elementen van het array s gebeurt met het algoritme, zoals behandeld op blz. 29 en met behulp van 4.34:

$$\begin{aligned} &n_*(s[1], ..., s[lengte-1], 2i-1) = a[n - lengte, energie - (2i-1)^2] \\ &energie = e_{max} - \textstyle\sum_{j=1}^{lengte-1} s[j]^2 \end{aligned} \tag{5.2}$$

De variabele energie is hierin weer de overgebleven energie na de bepaling van $n-1$ elementen. De variabele i wordt nu in een lus telkens met 1 opgehoogd. Er wordt begonnen met de waarde $i = 1$. De variabele *verminderdeindex* wordt berekend met:

$$verminderdeindex = index - a[n - lengte, energie - (2i-1)^2\,] \tag{5.3}$$

Als $verminderdeindex \geq 0$ is wordt de index:

$$index = verminderdeindex \tag{5.4}$$

en wordt de lus nog een keer doorlopen. Als $verminderdeindex < 0$ of wanneer $i = aantallevels$, a dus, (het bijbehorende element $s[lengte]$ heeft dan zijn maximale waarde $2a - 1$ bereikt en kan niet groter worden) wordt de lus gestopt. De variabele i bevat dan zijn juiste waarde en het element $s[lengte]$ is dan gelijk aan:

$$s[lengte] = 2i - 1 \tag{5.5}$$

$s[n]$ volgt ook alweer uit de waarde van $index$ zoals die na de bepaling van $n-1$ elementen overblijft:

$$s[n] = 2\,index + 1 \tag{5.6}$$

5.1.4 Beschrijving van de procedure decodeer.

In een lus wordt bij de variabele $index$ telkens de bijdrage van een element $s[lengte]$ voor $1 \leq lengte \leq n-1$ opgeteld. De bijdrage van $s[lengte]$ is gelijk aan:

$$\begin{array}{l} \sum_{i=1}^{\frac{s[lengte]-1}{2}} a[n - lengte, energie - (2i-1)^2] \\ energie = y - \sum_{j=1}^{lengte-1} s[j]^2 \end{array} \tag{5.7}$$

Volgens 4.36 wordt de bijdrage van $s[n]$:

$$\frac{s[n]-1}{2} \tag{5.8}$$

5.2 Het uittesten van de unit pam.

Aan het einde van de procedure init zijn weer schrijfopdrachten geplaatst die de getallen $a[x,y]$ op het scherm plaatsen. Deze getallen bleken te kloppen met de waarden in tabel 4.2. De procedure init blijkt dus goed te werken. Na deze controle zijn de schrijfopdrachten weer verwijderd.

Er is ook een programma pamtest geschreven om de procedures encodeer en decodeer uit te testen. Dit testprogramma staat in de file pamtest.pas op disk en de listing bevindt zich weer in de appendix. Dit testprogramma is qua opzet gelijk aan het testprogramma voor de unit energie in de file entest.pas. De uitvoer van pamtest is geschreven naar de file pamuit (beschikbaar op disk en in de appendix). Bij het programma pamtest zijn de indices van de voorbeeldrijtjes uit tabel 4.3 ingevoerd. Het codewoord dat door de procedure encodeer uit de index gevonden wordt blijkt te kloppen met het voorbeeldrijtje in de tabel dat bij deze index hoort. Ook blijkt dat de index die door decodeer weer teruggevonden wordt uit het codewoord gelijk te zijn aan de oorspronkelijke index die bij encodeer is ingevoerd. Na deze test mag dus aangenomen worden dat de unit pam goed werkt.

5.3 Het programma energieminimalisatie.

Met het programma energieminimalisatie, zich bevindend in de file minimal.pas en in de appendix kan men een optimale waarde vinden voor het aantal pamniveau's a_{bol} van het bolvormige stelsel waarvoor de gemiddelde energie minimaal wordt. Daarna berekent het programma de winst in gemiddelde energie die met het bolvormige stelsel behaald kan worden ten opzichte van het kubusvormige stelsel.

Het programma vraagt om de invoer van:

-*r*: het aantal te verzenden bits per dimensie.
-*n*: de dimensie.

Voor het aantal codewoorden *min* dat nodig is geldt:

$$min = 2^{rn} \tag{5.9}$$

Het aantal pamniveau's a_{kubus} voor een kubusvormig stelsel volgt uit het aantal benodigde codewoorden *min*:

$$a_{kubus}^{n} = 2^{rn} \rightarrow a_{kubus} = 2^{r} \tag{5.10}$$

De procedure aantalrijtjes moet er voor zorgen dat het aantal codewoorden in de bol *muit* minstens gelijk is aan *min*:

$$muit = A(n, e_{max}) \geq min \tag{5.11}$$

Omdat geldt:

$$A(n, e_{max}) \leq a_{bol}^{n} \tag{5.12}$$

volgt uit de v.g.l. 5.9 en 5.11:

$$a_{bol} \geq 2^{r} \rightarrow a_{bol} \geq a_{kubus} \tag{5.13}$$

Aan de gebruiker wordt nu telkens gevraagd een waarde voor a_{bol} in te voeren die dus moet voldoen aan 5.13. De procedure aantalrijtjes bepaalt de waarden voor *muit* en de maximale energie e_{max} van de codewoorden die nodig zijn om te voldoen aan 5.11. De procedure levert tevens het aantal codewoorden $B(v, w)$ ter lengte v en met constante energie w voor $1 \leq v \leq n$ en $0 \leq w \leq e_{max}$. Hiermee wordt op twee manieren de gemiddelde energie van de bol uitgerekend.

De gemiddelde energie $E_{bol,1}$ is alleen gebaseerd op het aantal codewoordem *min* in de bol die men nodig heeft en wordt gegeven door:

$$E_{bol,1} = \sum_{e=0}^{e_{max}-1} \frac{B(n,e)e}{min} + \frac{(min - muit + B(n, e_{max}))e_{max}}{min} \tag{5.14}$$

De tweede waarde voor de gemiddelde energie $E_{bol,2}$ is gebaseerd op alle codewoorden *muit* in de bol:

$$E_{bol,2} = \sum_{e=0}^{e_{max}} \frac{B(n,e)e}{muit} \tag{5.15}$$

a_{bol}	$muit$	$emax$	$E_{bol,1}$	$gain1$	$E_{bol,2}$	$gain2$
2	256	72	40		40	
3	451	48	36,3	1,10 of 0,43 dB	41,3	0,967 of -0,144 dB
4	451	48	36,3		41,3	

Tabel 5.1: Resultaten energieminimalisatie voor $r = 1$, $n = 8$, $a_{kubus} = 2$ en $min = 256$.

a_{bol}	$muit$	$emax$	$E_{bol,1}$	$gain1$	$E_{bol,2}$	$gain2$
4	65536	392	168		168	
5	71440	184	145,2		148,4	
6	77056	184	144,3	1,16 of 0,66 dB	150,2	1,12 of 0,49 dB
7	77120	184	144,3		150,2	
8	77120	184	144,3		150,2	

Tabel 5.2: Resultaten energieminimalisatie voor $r = 2$, $n = 8$, $a_{kubus} = 4$ en $min = 65536$.

De gebruiker kan nu telkens een waarde voor a_{bol} invoeren totdat voor een zekere waarde $E_{bol,1}$ en $E_{bol,2}$ hun minimale waarde bereikt hebben. Het programma berekent tot slot de winsten $gain1$ en $gain2$ van $E_{bol,1}$ en $E_{bol,2}$ ten opzichte van de gemiddelde energie van de kubus:

$$gain1 = \frac{n\frac{4a_{kubus}^2-1}{3}}{E_{bol,1}} \tag{5.16}$$

$$gain2 = \frac{n\frac{4a_{kubus}^2-1}{3}}{E_{bol,2}} \tag{5.17}$$

De resultaten van dit programma worden nu vermeld in de tabellen 5.1 tot en met 5.4.

5.4 Het programma simulatie.

Het programma simulatie, in de file simul.pas en in de appendix, genereert met behulp van de functie random een aantal indices. Van de met de procedure encodeer uit de unit pam gevonden codewoorden wordt de gemiddelde energie en de gain ten opzichte van de kubus uitgerekend. Deze waarden kan men vergelijken met datgene wat er gevonden is m.b.v.

a_{bol}	$muit$	$emax$	$E_{bol,1}$	$gain1$	$E_{bol,2}$	$gain2$
2	65536	144	80		80	
3	131027	80	68,3	1,17 of 0,687 dB	74,15	1,08 of 0,33 dB
4	132963	80	68,3		74,21	
5	132963	80	68,3		74,21	

Tabel 5.3: Resultaten energieminimalisatie voor $r = 1$, $n = 16$, $a_{kubus} = 2$ en $min = 65536$.

a_{bol}	$muit$	$emax$	$E_{bol,1}$	$gain1$	$E_{bol,2}$	$gain2$
2	16777216	216	120		120	
3	38431646	112	99,33		106,47	
4	39641102	112	99,17	1,21 of 0,828 dB	106,57	1,13 of 0,515 dB
5	39641678	112	99,17		106,57	
6	39641678	112	99,17		106,57	

Tabel 5.4: Resultaten energieminimalisatie voor $r = 1$, $n = 24$, $a_{kubus} = 2$ en $min = 16777216$.

het programma energieminimalisatie. Tevens wordt van elk codewoord de index weer terug bepaald met de procedure decodeer. Deze index wordt vergeleken met de oorspronkelijk met random gegenereerde index en wanneer deze ongelijk zijn wordt telkens de variabele *fouten* met 1 opgehoogd.

Aan het programma moeten de parameters r en n opgegeven worden, zoals bij het programma energieminimalisatie. Met 5.10 wordt a_{kubus} uitgerekend en 5.9 levert min. De gebruiker moet een waarde voor a_{bol} invoeren die voldoet aan 5.13. De procedure init van de unit pam bepaalt $muit$ en $emax$ met 5.11.

De gemiddelde energie van de bol kan weer op twee manieren bepaald worden door enerzijds uit te gaan van alleen min codewoorden en anderzijds uit te gaan van alle $muit$ codewoorden. In het eerste geval worden met de functie random $aantal1$ indices gegenereerd waarvan de waarde in het interval $[0, min-1]$ ligt. De parameter $aantal1$ moet door de gebruiker opgegeven worden. Voor elke index wordt de energie van het bijbehorende codewoord uitgerekend:

$$energie = \sum_{i=1}^{n} codewoord[i]^2 \tag{5.18}$$

De gemiddelde energie $E_{bol,1}$ volgt met:

$$E_{bol,1} = \frac{\sum_{aantal1} energie}{aantal1} \tag{5.19}$$

Hierna worden $aantal2$ indices met waarde liggend in het interval $[0, muit-1]$ gegenereerd. De gemiddelde energie $E_{bol,2}$ gebaseerd op alle codewoorden $muit$ in de bol is gelijk aan:

$$E_{bol,2} = \frac{\sum_{aantal2} energie}{aantal2} \tag{5.20}$$

De winsten $gain1$ en $gain2$ ten opzichte van de kubus volgen met de v.g.l. 5.16 en 5.17. De resultaten bevinden zich in de tabellen 5.5 tot en met 5.8. Er is genomen: $aantal1 = aantal2 = 10000$.

muit	*emax*	$E_{bol,1}$	*gain*1	$E_{bol,2}$	*gain*2	*fouten*
451	48	40,0	$-2,2\ 10^{-3}$ of 0,999 dB	41,3	0,97 of -0,14 dB	0

Tabel 5.5: Resultaten simulatie voor $r = 1$, $n = 8$, $a_{bol} = 3$ en $min = 256$.

muit	*emax*	$E_{bol,1}$	*gain*1	$E_{bol,2}$	*gain*2	*fouten*
77056	184	148,4	1,13 of 0,54 dB	150,3	1,12 of 0,48 dB	0

Tabel 5.6: Resultaten simulatie voor $r = 2$, $n = 8$, $a_{bol} = 6$ en $min = 65536$.

5.5 Bespreking van de resultaten van energieminimalisatie en simulatie.

Wanneer men de tabellen 5.1 tot en met 5.4 vergelijkt met de tabellen 5.5 tot en met 5.8 ziet men allereerst dat de programma's energieminimalisatie en simulatie dezelfde waarden voor *min*, *muit* en *emax* leveren. De waarden voor $E_{bol,2}$ die beide programma's berekenen stemmen ook met elkaar overeen zodat men kan concluderen dat beide programma's correct werken.

De waarden voor *gain*2 die beide programma's berekenen liggen toch wel duidelijk lager dan de theoretische waarden die in tabel 4.1 staan vermeld. Maar dat komt dus omdat *gain*2 gebaseerd is op alle codewoorden *muit* in de bol. Om er voor te zorgen dat $muit \geq min$ wordt de maximale energie *emax* telkens opgehoogd en dus worden er meer codewoorden van hogere energie erbij gehaald totdat aan deze voorwaarde voldaan is. Men heeft dus een aantal codewoorden $muit - min$ over die men niet nodig heeft maar die een hogere waarde voor de gemiddelde energie veroorzaken. Zoals men in de tabellen 5.1 en 5.5 kan zien blijkt *gain*2 voor het geval $r = 1$, $n = 8$ en $a_{bol} = 3$ zelfs kleiner dan 1 te zijn: je hebt hier dus geen winst. De gemiddelde energie is dus groter dan de gemiddelde energie voor het kubusvormige stelsel. Volgens v.g.l. 5.10 is het aantal codewoorden in de kubus precies gelijk aan *min*: er zijn bij de kubus geen overbodige codewoorden.

Met de gemiddelde energie $E_{bol,1}$ gebaseerd op *min* kan men inderdaad zoals blijkt uit de tabellen een winst *gain*1 bereiken die hoger is dan *gain*2. Men merkt op uit de tabellen dat de waarde voor *gain*1 bij het programma energieminimalisatie hoger is dan bij het simulatieprogramma. Dit komt omdat bij simulatie $E_{bol,1}$ op een andere manier bepaald wordt dan bij energieminimalisatie. Bij het simulatieprogramma worden de codewoorden met index liggend in het interval $[min, muit-1]$ weggelaten. De energie van deze codewoorden kan willekeurige waarden aannemen. Bij het energieminimalisatieprogramma worden alleen codewoorden met de maximale energie *emax* weggelaten, zoals blijkt uit v.g.l. 5.14.

muit	*emax*	$E_{bol,1}$	*gain*1	$E_{bol,2}$	*gain*2	*fouten*
131027	80	73,3	1,09 of 0,38 dB	74,1	1,08 of 0,33 dB	0

Tabel 5.7: Resultaten simulatie voor $r = 1$, $n = 16$, $a_{bol} = 3$ en $min = 65536$.

$muit$	$emax$	$E_{bol,1}$	$gain1$	$E_{bol,2}$	$gain2$	$fouten$
39641102	112	106	1,13 of 0,54 dB	106,5	1,126 of 0,52 dB	0

Tabel 5.8: Resultaten simulatie voor $r = 1$, $n = 24$, $a_{bol} = 4$ en $min = 16777216$.

Zo kan er een kleinere waarde voor $E_{bol,1}$ bereikt worden dan bij het simulatieprogramma.

Zoals blijkt uit de tabellen is het zeer aan te raden om alleen codewoorden met de maximale energie weg te laten omdat men dan een duidelijk hogere waarde voor $gain1$ kan bereiken die de theoretische waarde voor de winst goed benadert. De $gain1$ bereikt met simulatie blijkt niet echt veel hoger te zijn dan $gain2$. In de praktijk zou men dit kunnen realiseren door $muit - min$ indexwaarden uit het interval $[0, muit - 1]$ waarvan het codewoord de energie $emax$ heeft niet te gebruiken. Men moet elke indexwaarde gegenereerd door de bron uit het interval $[0, min - 1]$ omzetten naar een indexwaarde uit de verzameling van indices liggend in het interval $[0, muit - 1]$ zonder de verboden indexwaarden.

5.6 Het programma Gaussisch.

Het is zo dat de verdeling van de niveau's bij de bol Gaussisch moet zijn. Dit kan men als volgt aantonen:
Laat de vector $\mathbf{x}$ een punt aanduiden in de $n-$ dimensionale bol met volume:

$$V_{bol} = \frac{\pi^{n/2}}{(n/2)!}(nP)^{n/2} \tag{5.21}$$

Hierin stelt P de energie per dimensie voor. De entropie van $\mathbf{x}$ is gedefiniëerd als:

$$H(\mathbf{x}) = \int_{bol} -p(\mathbf{x}) \ln p(\mathbf{x})\, d\mathbf{x} \tag{5.22}$$

De verdeling van $\mathbf{x}$ over de bol is uniform:

$$p(\mathbf{x}) = \frac{1}{V_{bol}} \tag{5.23}$$

Dus de entropie wordt:

$$H(\mathbf{x}) = \ln V_{bol} \tag{5.24}$$

Voor de entropie van de vector $\mathbf{x}$ per dimensie krijgen we met de vergelijkingen 5.21 en 5.24:

$$\frac{H(\mathbf{x})}{n} = \frac{1}{n} \ln \frac{\pi^{n/2}(nP)^{n/2}}{(n/2)!}$$

Met de benadering van Stirling zoals we die gezien hebben in v.g.l. 4.29 wordt dit voor grote n:

$$\frac{H(\mathbf{x})}{n} = 1/2 \ln (2\pi e P) \tag{5.25}$$

Uit de entropie van $\mathbf{x}$ per dimensie volgt een ongelijkheid voor de entropie van elk element van $\mathbf{x}$:

$$\frac{H(\mathbf{x})}{n} = \frac{H(x_1) + H(x_2 \mid x_1) + \ldots + H(x_n \mid x_1 \ldots x_{n-1})}{n}$$

$$\frac{H(\mathbf{x})}{n} \leq \frac{H(x_1) + H(x_2) + \ldots + H(x_n)}{n} = H(x)$$

We hebben nu de volgende ongelijkheid voor $H(x)$:

$$1/2 \ln (2\pi e P) \leq H(x) \tag{5.26}$$

We kunnen nog een ongelijkheid voor $H(x)$ vinden door gebruik te maken van theorema 7.4.1. op pagina 335 in [9]. Deze zegt dat voor de entropie

$$H(x) = -\int_{-\infty}^{\infty} p(x) \ln p(x)\, dx$$

van een kansverdeling $p(x)$ die voldoet aan:

$$\overline{x^2} = A$$

geldt:

$$H(x) \leq 1/2 \ln(2\pi e A) \tag{5.27}$$

De maximale waarde voor $H(x)$ in v.g.l. 5.27 wordt alleen bereikt als $p(x)$ Gaussisch is met variantie:

$$\sigma^2 = A \tag{5.28}$$

Omdat voor elk element x geldt:

$$\overline{x^2} \leq P$$

volgt met v.g.l. 5.27:

$$H(x) \leq 1/2 \ln(2\pi e P) \tag{5.29}$$

Uit de ongelijkheden 5.26 en 5.29 volgt dat bij benadering voor grote n:

$$H(x) \approx 1/2 \ln(2\pi e P) \tag{5.30}$$

Volgens het theorema moet bij grote n de verdeling voor x dus Gaussisch zijn met variantie:

$$\sigma^2 = P \tag{5.31}$$

Om deze uitspraak te verifiëren is er het programma Gaussisch geschreven, beschikbaar in de file Gaus.pas en in de appendix. Dit programma telt voor elk niveau uit het alfabet $\{1, 3, 5, ..., 2a-1\}$ hoe vaak het voorkomt in de totale verzameling van codewoorden en berekent hiervan de fractie, gebaseerd op *min* en *muit*:

$$fractie_{min}(niveau) = \frac{\sum_{min\ codewoorden} aantal\ keren\ in\ codewoord}{min\, n} \tag{5.32}$$

$$fractie_{muit}(niveau) = \frac{\sum_{muit\ codewoorden} aantal\ keren\ in\ codewoord}{muit\, n} \tag{5.33}$$

Deze fracties worden door het programma op het scherm geschreven. Deze fracties behoren Gaussisch verdeeld te zijn met standaarddeviaties:

$$\sigma_1 = \sqrt{\frac{E_{bol,1}}{n}} \tag{5.34}$$

$$\sigma_2 = \sqrt{\frac{E_{bol,2}}{n}} \tag{5.35}$$

De fracties worden ook berekend uit de formule voor de Gaussische verdeling:

$$fractie_{min}(niveau\ i) = \int_{i-1}^{i+1} \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{x^2}{2\sigma_1^2}}\, dx \tag{5.36}$$

$$fractie_{muit}(niveau\ i) = \int_{i-1}^{i+1} \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{x^2}{2\sigma_2^2}}\, dx \tag{5.37}$$

Dit kan men schrijven als:

$$fractie_{min}(i) = 2\left(P\left(\frac{i+1}{\sigma_1}\right) - P\left(\frac{i-1}{\sigma_1}\right)\right) \tag{5.38}$$

$$fractie_{muit}(i) = 2\left(P\left(\frac{i+1}{\sigma_2}\right) - P\left(\frac{i-1}{\sigma_2}\right)\right) \tag{5.39}$$

De functie $P(x)$ is gedefinieerd door:

$$P(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}\, dt \tag{5.40}$$

Volgens [8] kan men deze functie benaderen met:

$$P(x) = 1 - Z(x)(b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + b_5 t^5) + \epsilon(x) \tag{5.41}$$

met:

$$t = \frac{1}{1+px} \tag{5.42}$$

$$| \epsilon(x) | < 7,5\ 10^{-8} \tag{5.43}$$

$$\begin{array}{ll} p = 0,2316419 & b_3 = 1,781477937 \\ b_1 = 0,319381530 & b_4 = -1,821255978 \\ b_2 = -0,356563782 & b_5 = 1,330274429 \end{array} \tag{5.44}$$

De waarden voor de fracties gevonden met de v.g.l. 5.38 en 5.39 worden door het programma op het scherm geschreven. De gebruiker kan nu nagaan of deze waarden overeen komen met de waarden gevonden in 5.32 en 5.33. De uitvoer van Gaussisch is omgeleid naar de file Gausuit (zie de appendix). Wanneer men de resultaten vergelijkt kan men inderdaad verifiëren dat de fracties van de niveau's een Gaussische verdeling aannemen.

Hoofdstuk 6

Conclusies.

Het is gebleken dat de enumeratieve coderingsmethode succesvol kan worden toegepast bij het ontwerpen van codes waarvan de codewoorden een bepaalde eigenschap hebben mits men een uitdrukking kan vinden voor het aantal van deze codewoorden. Voor de energiebegrensde code in deze stage is geen expliciete formule gevonden waarmee het aantal codewoorden uitgerekend kan worden. In plaats daarvan is er wel een zeer handig en robust algoritme gevonden waarmee een computerprogramma probleemloos de aantallen $A(v, w_{max})$ kan bepalen.

Met de energiebegrensde code bleek het mogelijk te zijn om een bolvormig signaalpuntenstelsel voor PAM in de praktijk te realiseren. In [6] wordt alleen aangeduid dat zo'n stelsel theoretisch de minste gemiddelde energie heeft, maar er wordt niet aangegeven hoe dit stelsel in de praktijk gerealiseerd kan worden.

Volgens de resultaten in de tabellen 5.1 tot en met 5.4 en 5.5 tot en met 5.8 kan er met behulp van de energiebegrensde code inderdaad een winst bereikt worden ten opzichte van de kubus. Deze winst blijkt echter niet echt hoog te zijn: rond de 1,1. Het programma energieminimalisatie laat duidelijk zien dat men een betere winst kan verkrijgen die de theoretische winst behandeld in artikel [6] goed benadert. Men moet dan een systeem programmeren waarbij alleen een vaste verzameling van *min* indexwaarden uit het interval $[0, muit - 1]$ gebruikt wordt en de $muit - min$ andere indexwaarden met energie *emax* voor het codewoord verboden zijn.

Bibliografie

[1] Schalkwijk, J.P.M. Informatietheorie 1. Faculteit Elektrotechniek.
Technische Universiteit Eindhoven, mei 1981. Dictaatnr. 5514.

[2] Cover, T.M. Enumerative source encoding.
IEEE Trans. Inf. Theory. (USA) , Vol. 19(1973), No. 1, p. 73-77.

[3] Lynch, T.J. Sequence time coding for data compression.
Proc. IEEE (USA) , Vol. 54(1966), No. 10, p. 1490-1491.

[4] Schalkwijk, J.P.M. An algorithm for source coding.
IEEE Trans. Inf. Theory. (USA) , Vol. 18(1972), No. 3, p. 395-399.

[5] Schalkwijk, J.P.M. Coding by lexicographical enumeration.
Nederlands Elektronica- en Radiogenootschap., Vol. 39(1974), No. 5-6, p. 163-167.

[6] Forney, G.D. et al. Efficient modulation for band-limited channels.
IEEE journal on selected areas in communications,
Vol. SAC-2(1984), No. 5, p. 632-646.

[7] Schalkwijk, J.P.M. Communicatieprincipes. Faculteit Elektrotechniek.
Technische Universiteit Eindhoven, april 1989. Dictaatnr. 5539.

[8] Abramowitz, M. and I.A. Stegun.
Handbook of mathematical functions with formules,
graphs and mathematical tables, New York: Dover 1970.

[9] Gallager, R.G.
Information theory and reliable communication.

New York: Wiley 1968.

Appendix.

De unit energie in de file energie.pas.

```
unit energie; { j.j. wuijts.
  Deze unit bevat procedures voor het coderen van een
  lexicografische index naar zijn bijbehorende
  energiebegrensde codewoord x1,x2,...,xn
  waarvoor geldt dat elke xi een waarde kan aannemen
  uit het alfabet (0,1,2,...,k)
  en waarvoor geldt dat x1*x1+x2*x2+...xn*xn<=emax.

  Voordat men gaat coderen en/of decoderen moet men
  de procedure init aanroepen.
  Aan deze procedure moet men als invoerparameters opgeven:
  kin: de laatste letter uit het alfabet (maximaal 22).
  nin: de lengte van het codewoord (maximaal 30).
  aantalindicesin: het aantal verschillende waarden die de
  index die door de bron geleverd wordt en gecodeerd moet
  worden kan aannemen.

  De procedure geeft als uitvoerparameter terug:
  aantalindicesuit: het aantal verschillende indexwaarden
  die de code aankan. De procedure init zorgt ervoor
  dat aantalindicesuit>=aantalindicesin.
  emaxuit: de maximale energie die nodig is om
  ervoor te zorgen dat aantalindicesuit>=aantalindicesin.

  De procedure encodeer verzorgt het coderen van de index naar
  een energiebegrensd codewoord. Men voert een index in via de
  invoerparameter indexin en krijgt als uitvoerparameter een
  array s terug dat het codewoord bevat. De gebruiker moet in
  zijn programma het array dat hij gebruikt voor het codewoord
  van het type symboolrij declareren, dus bijvoorbeeld var
  codewoord:symboolrij. In de elementen s[1] tot en met s[n]
  staan de elementen x1 tot en met xn van het codewoord. De
  procedure decodeer decodeert het codewoord weer terug naar de
  index. Men geeft als invoerparameter een array s met het
  codewoord en ontvangt via de uitvoerparameter indexuit de
  index. }


 interface
 type symboolrij=array[1..30] of longint;
 procedure init(kin,nin,aantalindicesin :longint;
                var aantalindicesuit,emaxuit :longint);
 procedure encodeer(indexin:longint; var s:symboolrij);
 procedure decodeer(s:symboolrij; var indexuit:longint);

 implementation

 var k,n,i,y:longint;
     a:array[1..30,0..500] of longint;

 procedure init(kin,nin,aantalindicesin:longint;
                var aantalindicesuit,emaxuit:longint);

    var l,aantalindices,x:longint;
        kwadraat:array[0..500] of longint;
        eindekwadraten:boolean;

    begin
```

```
    k:=kin;n:=nin;
    for i:=0 to k do kwadraat[i]:=i*i;
    l:=-1;eindekwadraten:=false;
    aantalindices:=0;
    y:=-1;
    while aantalindices< aantalindicesin do
    begin
          y:=y+1;
          if eindekwadraten=false then
          begin
                if kwadraat[l+1]<=y then l:=l+1;
                if l=k then eindekwadraten:=true;
          end;
          a[1,y]:=l+1;
          for x:=2 to n do
          begin
                a[x,y]:=0;
                if l<>-1 then for i:=0 to l do
                a[x,y]:=a[x,y]+a[x-1,y-kwadraat[i]];;
          end;
          aantalindices:=a[n,y];
     end;
     emaxuit:=y;
     aantalindicesuit:=aantalindices;
   end;

procedure encodeer(indexin:longint; var s:symboolrij);

   var index,energie,lengte,verminderdeindex:longint;

   begin

     index:=indexin;energie:=y;
     for lengte:=1 to n-1 do
     begin
          s[lengte]:=-1;
          verminderdeindex:=index;
          while ( verminderdeindex>=0 )and( s[lengte]<>k ) do
          begin
                index:=verminderdeindex;
                s[lengte]:=s[lengte]+1;
                if s[lengte]<>k then
                     verminderdeindex:=index-
                     a[n-lengte,energie-s[lengte]*s[lengte]];
          end;
          energie:=energie-s[lengte]*s[lengte];
      end;
      s[n]:=index;

    end;

 procedure decodeer(s:symboolrij; var indexuit:longint);

    var index,energie,lengte:longint;

    begin

      energie:=y;index:=0;
      for lengte:=1 to n-1 do
      begin
           if s[lengte]>0 then for i:=0 to s[lengte]-1 do
           index:=index+a[n-lengte,energie-i*i];;
           energie:=energie-s[lengte]*s[lengte];
      end;
```

```
        index:=index+s[n];
        indexuit:=index;

      end;

end.
```

Het testprogramma test in de file entest.pas.

```
program test;
{ j.j. Wuijts.
  Dit programma test de werking van de unit energie
  voor het geval:
  alfabet: (0,1,2).
  lengte van de codewoorden: 7.
  het aantal verschillende indexwaarden geleverd
  door de bron: 912.
  Men geeft een index op aan het programma.
  Het bijbehorende codewoord wordt bepaald met
  encodeer en naar het scherm geschreven.
  De index wordt weer teruggevonden met decodeer
  en ook op het scherm geplaatst.
}

 uses energie;
 var muit,emax,index,tel:longint;
     codewoord:symboolrij;
     antwoord:char;
 begin
       init(2,7,912,muit,emax);
       writeln('muit: ',muit);
       writeln('emax: ',emax);
       repeat
              writeln('geef index');
              readln(index);
              writeln('ingevoerde index: ',index);
              encodeer(index,codewoord);
              write('het codewoord: ');
              for tel:=1 to 7 do write(codewoord[tel]);
              writeln;
              index:=0;
              decodeer(codewoord,index);
              writeln('gedecodeerde index: ',index);
              writeln('stoppen j/n?');
              readln(antwoord);
       until antwoord='j';
 end.
```

De testresultaten van het programma test in de file enuit.

```
muit: 912
emax: 10
geef index
ingevoerde index: 15
het codewoord: 0000120
gedecodeerde index: 15
stoppen j/n?
geef index
```

```
ingevoerde index: 16
het codewoord: 0000121
gedecodeerde index: 16
stoppen j/n?
geef index
ingevoerde index: 17
het codewoord: 0000122
gedecodeerde index: 17
stoppen j/n?
geef index
ingevoerde index: 18
het codewoord: 0000200
gedecodeerde index: 18
stoppen j/n?
geef index
ingevoerde index: 19
het codewoord: 0000201
gedecodeerde index: 19
stoppen j/n?
geef index
ingevoerde index: 20
het codewoord: 0000202
gedecodeerde index: 20
stoppen j/n?
geef index
ingevoerde index: 21
het codewoord: 0000210
gedecodeerde index: 21
stoppen j/n?
geef index
ingevoerde index: 22
het codewoord: 0000211
gedecodeerde index: 22
stoppen j/n?
geef index
ingevoerde index: 23
het codewoord: 0000212
gedecodeerde index: 23
stoppen j/n?
geef index
ingevoerde index: 561
het codewoord: 1021010
gedecodeerde index: 561
stoppen j/n?
geef index
ingevoerde index: 562
het codewoord: 1021011
gedecodeerde index: 562
stoppen j/n?
geef index
ingevoerde index: 563
het codewoord: 1021020
gedecodeerde index: 563
stoppen j/n?
geef index
ingevoerde index: 564
het codewoord: 1021100
gedecodeerde index: 564
stoppen j/n?
geef index
ingevoerde index: 565
het codewoord: 1021101
gedecodeerde index: 565
stoppen j/n?
geef index
```

```
ingevoerde index: 566
het codewoord: 1021110
gedecodeerde index: 566
stoppen j/n?
geef index
ingevoerde index: 567
het codewoord: 1021111
gedecodeerde index: 567
stoppen j/n?
```

De unit pam in de file pam.pas.

```
unit pam; { j.j. wuijts.
  deze unit bevat procedures voor het coderen van een
  lexicografische index naar zijn bijbehorende
  energiebegrensde codewoord x1,x2,...,xn
  waarvoor geldt dat elke xi een waarde kan aannemen
  uit het alfabet (1,3,5,....,2a-1)
  en waarvoor geldt dat x1*x1+x2*x2+....+xn*xn<=emax.
  De niveau's (1,3,5,....,2a-1) worden standaard gebruikt
  bij pulse amplitude modulatie om digitale informatie over
  een kanaal te zenden. Het getal a staat voor het aantal
  niveau's en bepaalt tevens het maximale niveau 2a-1.

  Voordat men gaat coderen en/of decoderen moet men
  de procedure init aanroepen.
  Aan deze procedure moet men als invoerparameters opgeven:
  aantallevelsin: het aantal te gebruiken niveau's a (maximaal 11).
  nin: de lengte n van het codewoord (maximaal 30).
  aantalindicesin: het aantal verschillende waarden die
  de index die door de bron aangeleverd wordt en
  gecodeerd moet worden kan aannemen.
  De procedure geeft als uitvoerparameter terug:
  aantalindicesuit: het aantal verschillende indexwaarden
  die de code aankan. De procedure init zorgt ervoor dat
  aantalindicesuit>=aantalindicesin.
  emaxuit: de maximale energie emax die nodig is om
  ervoor te zorgen dat aantalindicesuit>=aantalindicesin.

  De procedure encodeer verzorgt het coderen van de index
  naar zijn bijbehorende energiebegrensde codewoord.
  Men geeft op als invoerparameter indexin de index en
  krijgt als uitvoerparameter de array s terug die
  het codewoord bevat.
  In de elementen s[1] tot en met s[n] staan de elementen
  x1 tot en met xn van het codewoord.
  De gebruiker moet in zijn programma het array dat hij
  gebruikt voor het codewoord van het type symboolrij
  declareren, dus bijvoorbeeld var codewoord:symboolrij.

  De procedure decodeer decodeert het codewoord weer
  terug naar de index.
  Men geeft als invoerparameter een array s met het
  codewoord en ontvangt via de uitvoerparameter indexuit
  de index.  }


 interface
 type symboolrij=array[1..30] of longint;
 procedure init(aantallevelsin,nin,aantalindicesin:longint;
                var aantalindicesuit,emaxuit:longint);
 procedure encodeer(indexin:longint;var s:symboolrij);
```

```
procedure decodeer(s:symboolrij;var indexuit:longint);

implementation

var aantallevels,n,i,y:longint;
    a:array[1..30,0..500] of longint;

procedure init(aantallevelsin,nin,aantalindicesin:longint;
               var aantalindicesuit,emaxuit:longint);

   var l,aantalindices,x:longint;
       kwadraat:array[0..500] of longint;
       eindekwadraten: boolean;

   begin

      aantallevels:=aantallevelsin;
      n:=nin;
      for i:=1 to aantallevels do kwadraat[i]:=(2*i-1)*(2*i-1);
      l:=0; eindekwadraten:=false;
      aantalindices:=0;
      y:=-1;
      while aantalindices< aantalindicesin do
      begin
            y:=y+1;
            if eindekwadraten=false then
            begin
                  if kwadraat[l+1]<=y then l:=l+1;
                  if l=aantallevels then eindekwadraten:=true;
            end;
            a[1,y]:=l;
            for x:=2 to n do
            begin
                  a[x,y]:=0;
                  if l<>0 then for i:=1 to l do
                  a[x,y]:=a[x,y]+a[x-1,y-kwadraat[i]];;
            end;
            aantalindices:=a[n,y];
      end;
      emaxuit:=y;
      aantalindicesuit:=aantalindices;
   end;

procedure encodeer(indexin:longint;var s:symboolrij);

   var energie,index,lengte,verminderdeindex:longint;

   begin
       index:=indexin; energie:=y;
       for lengte:=1 to n-1 do
       begin
           i:=0;verminderdeindex:=index;
           while (verminderdeindex>=0) and (i<>aantallevels) do
           begin
                 index:=verminderdeindex;
                 i:=i+1;
                 if i<>aantallevels then
                           verminderdeindex:=index-
                           a[n-lengte,energie-(2*i-1)*(2*i-1)];
           end;
           energie:=energie-(2*i-1)*(2*i-1);
           s[lengte]:=2*i-1;
       end;
       s[n]:=2*index+1;
   end;
```

```
  procedure decodeer(s:symboolrij;var indexuit:longint);

     var energie,index,lengte:longint;

     begin
         energie:=y;index:=0;
         for lengte:=1 to n-1 do
         begin
            if s[lengte]>1 then for i:=1 to (s[lengte]-1)div 2 do
            index:=index+a[n-lengte,energie-(2*i-1)*(2*i-1)];;
            energie:=energie-s[lengte]*s[lengte];
         end;
         index:=index+(s[n]-1)div 2;
         indexuit:=index;
     end;


  end.
```

Het testprogramma pamtest in de file pamtest.pas.

```
program pamtest;

{ j.j. Wuijts.
  Met dit programma kan men eenvoudig de werking
  van de unit pam uittesten voor het geval:
  alfabet: (1,3,5).
  lengte codewoord:4.
  aantal verschillende indices gevraagd
  door de bron:32.
  Het programma vraagt de gebruiker om een index,
  codeert deze met de procedure encodeer, schrijft
  het codewoord op het scherm, decodeert het codewoord
  weer terug naar de index en schrijft deze indexwaarde
  tenslotte ook op het scherm.
 }

 uses pam;
 var muit,emax,index,tel:longint;
     codewoord:symboolrij;
     antwoord:char;
 begin
       init(3,4,32,muit,emax);
       writeln('muit: ',muit);
       writeln('emax: ',emax);
       repeat
              writeln('geef index');
              readln(index);
              writeln('ingevoerde index: ',index);
              encodeer(index,codewoord);
              write('het codewoord: ');
              for tel:=1 to 4 do write(codewoord[tel]);
              writeln;
              index:=0;
              decodeer(codewoord,index);
              writeln('gedecodeerde index: ',index);
              writeln('stoppen j/n?');
              readln(antwoord);
       until antwoord='j';
end.
```

De resultaten van het testprogramma pamtest in de file pamuit.

```
muit: 32
emax: 36
geef index
ingevoerde index: 0
het codewoord: 1111
gedecodeerde index: 0
stoppen j/n?
geef index
ingevoerde index: 1
het codewoord: 1113
gedecodeerde index: 1
stoppen j/n?
geef index
ingevoerde index: 2
het codewoord: 1115
gedecodeerde index: 2
stoppen j/n?
geef index
ingevoerde index: 6
het codewoord: 1151
gedecodeerde index: 6
stoppen j/n?
geef index
ingevoerde index: 7
het codewoord: 1153
gedecodeerde index: 7
stoppen j/n?
geef index
ingevoerde index: 8
het codewoord: 1311
gedecodeerde index: 8
stoppen j/n?
geef index
ingevoerde index: 9
het codewoord: 1313
gedecodeerde index: 9
stoppen j/n?
geef index
ingevoerde index: 10
het codewoord: 1315
gedecodeerde index: 10
stoppen j/n?
geef index
ingevoerde index: 11
het codewoord: 1331
gedecodeerde index: 11
stoppen j/n?
geef index
ingevoerde index: 12
het codewoord: 1333
gedecodeerde index: 12
stoppen j/n?
geef index
ingevoerde index: 13
het codewoord: 1351
gedecodeerde index: 13
stoppen j/n?
geef index
ingevoerde index: 14
het codewoord: 1511
```

```
gedecodeerde index: 14
stoppen j/n?
geef index
ingevoerde index: 15
het codewoord: 1513
gedecodeerde index: 15
stoppen j/n?
geef index
ingevoerde index: 31
het codewoord: 5311
gedecodeerde index: 31
stoppen j/n?
```

Het programma energieminimalisatie in de file minimal.pas.

```
program energieminimalisatie;

{ j.j. Wuijts.
  Met behulp van dit programma kan de gebruiker een optimale
  waarde vinden voor het aantal niveau's abol van het
  n-dimensionale bolvormige stelsel waarvoor de gemiddelde
  energie van de codewoorden minimaal is. Vervolgens wordt
  de winst in energie berekend van het bolvormige stelsel
  ten opzichte van het kubusvormige stelsel.

  Met programma vraagt om de invoer van twee gegevens:

  - r: het aantal te verzenden bits per dimensie.
  - n: de dimensie.
                                                    rn
  Daarna wordt hieruit het aantal codewoorden min=2
  berekend dat nodig is. Met aantal pamniveau's akubus dat
  nodig is voor de kubus wordt bepaald uit:
        n  rn                r
  akubus =2  . Dus: akubus=2 .

  Met programma vraagt nu in een lus telkens om de
  invoer van het aantal pamniveau's abol voor de bol.
  Voor abol moet gelden:

  abol>=akubus.

  De maximale energie en het aantal codewoorden
  muit=A(n,emax) van de bol waarvoor geldt:

  muit=A(n,emax)>=min worden bepaald.

  De gemiddelde energie van alleen het aantal
  codewoorden min dat nodig is en de gemiddelde
  energie van alle codewoorden muit in de bol
  wordt berekend.

  De gebruiker kan nu telkens nieuwe waarden voor abol
  invoeren en zo een optimale waarde vinden waarvoor
  de gemiddelde energie minimaal is.

  Tot slot wordt dan de gain, de winst, in de gemiddelde
  energie van de bol ten opzichte van de kubus voor zowel
  min als muit berekend.

  Met is duidelijk dat als men de gemiddelde energie
  alleen op het aantal codewoorden min die nodig zijn
```

```
  baseert men een hogere gain kan bereiken dan wanneer
  men uitgaat van alle codewoorden muit in de bol.

   }


type
    barray=array[1..30,0..500] of longint;

var
    i,r,n,abol,akubus,min,muit,emax,e:longint;
    gemenergiebol1,gemenergiebol2,gain1,gain2:real;
    antwoord:char;
    b:barray;


procedure aantalrijtjes(ain,nin,min:longint;
                        var muit,emaxuit:longint;var buit:barray);

var
   i,l,x,y:longint;
   kwadraat:array[0..500] of longint;
   eindekwadraten:boolean;

begin
   for i:=1 to ain do kwadraat[i]:=(2*i-1)*(2*i-1);
   l:=0;eindekwadraten:=false;
   muit:=0;y:=-1;
   while muit<min do
   begin
      y:=y+1;
      buit[1,y]:=0;
      if eindekwadraten=false then
      begin
         if kwadraat[l+1]<=y then begin
                                    l:=l+1;
                                    buit[1,y]:=1;
                                  end;
         if l=ain then eindekwadraten:=true;
      end;
      for x:=2 to nin do
      begin
         buit[x,y]:=0;
         if l<>0 then for i:=1 to l do
            buit[x,y]:=buit[x,y]+buit[x-1,y-kwadraat[i]];;
      end;
      muit:=0;
      for i:=0 to y do muit:=muit+buit[nin,i];
   end;
   emaxuit:=y;
end;



begin
   repeat
       writeln('geef waarden voor r en n.');
       writeln('het produkt rn mag niet groter zijn');
       writeln('dan 30');
       writeln('geef r');
       readln(r);
       writeln('geef n');
       readln(n);
```

```
 until n*r<=30;
 akubus:=1;
 for i:=1 to r do akubus:=akubus*2;
 min:=1;
 for i:=1 to n do min:=min*akubus;
 repeat
     abol:=0;
     while abol<akubus do
     begin
        writeln('geef aantal pamniveaus abol');
        writeln('voor de bol.');
        writeln('deze moet minimaal gelijk zijn aan');
        writeln('het aantal voor de kubus akubus:',akubus);
        readln(abol);
     end;
     aantalrijtjes(abol,n,min,muit,emax,b);
     writeln('r:',r);
     writeln('n:',n);
     writeln('akubus:',akubus);
     writeln('abol:',abol);
     writeln('min:',min);
     writeln('muit:',muit);
     writeln('emax:',emax);
     gemenergiebol1:=0;
     for e:=0 to emax-1 do
     begin
        gemenergiebol1:=gemenergiebol1+(b[n,e]/min)*e;
     end;
     gemenergiebol1:=gemenergiebol1+((min-muit+b[n,emax])/min)*emax;
     gemenergiebol2:=0;
     for e:=0 to emax do
     begin
        gemenergiebol2:=gemenergiebol2+(b[n,e]/muit)*e;
     end;

     writeln('gemiddelde energie1 van de bol bij gebruik');
     writeln('van min codewoorden:',gemenergiebol1);
     writeln('gemiddelde energie2 van de bol bij gebruik');
     writeln('van alle codewoorden muit',gemenergiebol2);
     writeln('nog een waarde voor abol? j/n');
     repeat
         readln(antwoord);
     until (antwoord='j') or (antwoord='n');
  until antwoord='n';
  gain1:=n*(4*akubus*akubus-1)/(3*gemenergiebol1);
  gain2:=n*(4*akubus*akubus-1)/(3*gemenergiebol2);
  writeln('gain1 bij gem energie1 :',gain1);
  gain1:=10*ln(gain1)/ln(10);
  writeln('gain1 in db: ',gain1);
  writeln('gain2 bij gem energie2 :',gain2);
  gain2:=10*ln(gain2)/ln(10);
  writeln('gain2 in db: ',gain2);
end.
```

Het programma simulatie in de file simul.pas.

```
program simulatie;

{ j.j.wuijts. Dit programma genereert eerst met de functie
  random aantal1 keer een index uit het interval [0,min-1].
  De bijbehorende codewoorden worden met de procedure encodeer
  uit de unit pam bepaald. De gemiddelde energie
```

```
  van deze codewoorden wordt berekend. Met de procedure decodeer
  wordt de index terugbepaald uit elk codewoord.
  Dan wordt het aantal fouten, dus het aantal keren dat
  de met decodeer teruggevonden index ongelijk is aan de
  oorspronkelijk bij encodeer ingevoerde waarde, geteld als
  deze mochten optreden.
  In het tweede geval wordt aantal2 keer met random een index
  uit het interval [0,muit-1] gekozen. Ook hier wordt de
  gemiddelde energie van de codewoorden berekend en als er
  fouten zijn worden deze geteld en toegevoegd aan het aantal
  fouten dat in het eerste geval is gevonden.
  Van de twee waarden voor de gemiddelde energie wordt de winst,
  de gain, berekend ten opzichte van de gemiddelde energie van
  de kubusvormige configuratie.
  De waarden voor de gemiddelde energie en de gain,
  gebaseerd op min en muit codewoorden, kan men
  vergelijken met de waarden zoals die gevonden zijn met het
  programma energieminimalisatie.

  In te voeren parameters:

  -r: aantal te verzenden bits per dimensie.
  -n: de dimensie.

  Het aantal codewoorden dat nodig is voor de bron wordt
  gegeven door:

       rn
  min=2  .

  -abol: het aantal pamniveaus van de bolvormige configuratie.
  -aantal1.
  -aantal2.

  Er wordt bepaald hoeveel maximale energie emax nodig is
  om er voor te zorgen dat het aantal codewoorden muit in
  de bol minstens gelijk is aan het gevraagde aantal min:

  muit=A(n,emax)>=min.

  }


uses pam;

var
    r,n,akubus,i,min,muit,abol,emax,fouten:longint;
    index,indexc,energie:longint;
    aantal1,aantal2,tel:longint;
    ensom,gemenergiebol1,gemenergiebol2:real;
    gain1,gain2:real;
    codewoord:symboolrij;

begin
   repeat
       writeln('geef waarden voor r en n.');
       writeln('het produkt rn mag niet groter zijn');
       writeln('dan 30');
       writeln('geef r');
       readln(r);
       writeln('geef n');
       readln(n);
   until n*r<=30;
   akubus:=1;
   for i:=1 to r do akubus:=akubus*2;
```

```
min:=1;
for i:=1 to n do min:=min*akubus;
abol:=0;
while abol<akubus do
begin
   writeln('in te voeren waarde voor abol moet');
   writeln('minimaal zijn:',akubus);
   readln(abol);
end;
writeln('geef een waarde voor aantal1');
readln(aantal1);
writeln('geef een waarde voor aantal2');
readln(aantal2);
init(abol,n,min,muit,emax);
ensom:=0;
fouten:=0;
for tel:=1 to aantal1 do
begin
      index:=trunc(min*random);
      encodeer(index,codewoord);
      energie:=0;
      for i:=1 to n do
      energie:=energie+codewoord[i]*codewoord[i];
      ensom:=ensom+energie;
      decodeer(codewoord,indexc);
      if indexc<>index then fouten:=fouten+1;
end;
gemenergiebol1:=ensom/aantal1;
ensom:=0;
for tel:=1 to aantal2 do
begin
      index:=trunc(muit*random);
      encodeer(index,codewoord);
      energie:=0;
      for i:=1 to n do
          energie:=energie+codewoord[i]*codewoord[i];
      ensom:=ensom+energie;
      decodeer(codewoord,indexc);
      if indexc<>index then fouten:=fouten+1;
end;
gemenergiebol2:=ensom/aantal2;
writeln('ingevoerde r:',r);
writeln('ingevoerde n:',n);
writeln('ingevoerde abol:',abol);
writeln('ingevoerde aantal1:',aantal1);
writeln('ingevoerde aantal2:',aantal2);
writeln('min:',min);
writeln('muit:',muit);
writeln('maximale energie emax:',emax);
writeln('gemiddelde energie1 van de random');
writeln('uit een set van min gegenereerde');
writeln('codewoorden:',gemenergiebol1);
writeln('gemiddelde energie2 met de set');
writeln('van alle muit codewoorden:',gemenergiebol2);
writeln('aantal fouten:',fouten);
gain1:=n*(4*akubus*akubus-1)/(3*gemenergiebol1);
gain2:=n*(4*akubus*akubus-1)/(3*gemenergiebol2);
writeln('gain1 bij gem energie1:',gain1);
gain1:=10*ln(gain1)/ln(10);
writeln('gain1 in db: ',gain1);
writeln('gain2 bij gem energie2:',gain2);
gain2:=10*ln(gain2)/ln(10);
writeln('gain2 in db:',gain2);
end.
```

Het programma Gaussisch in de file gaus.pas.

```
program gaussisch;

{ j.j.wuijts.
  Dit programma telt hoe vaak elke letter uit het alfabet
  (1,3,5,...,2abol-1) voorkomt in alle codewoorden bij elkaar.
  Er worden twee situaties apart beschouwd, een waarbij
  uitgegaan wordt van min codewoorden en een waarbij uitgegaan
  wordt van alle muit codewoorden.
  Dan wordt voor elke letter in het eerste geval
  de fractie1 hiervan berekend door het aantal door min*n te delen
  en in het tweede geval de fractie2 door het aantal door muit*n
  te delen. De fracties worden opgeslagen in de array's
  fracties1:array[1..abol] of real
  en fracties2:array[1..abol] of real.
  Wanneer men deze fracties in een grafiek plaatst moeten ze een
  gaussische verdeling aannemen met standaarddeviatie gelijk aan
  de gemiddelde energie van de codewoorden per dimensie.
  De gemiddelde energie wordt in beide gevallen door het
  programma uitgerekend waaruit dus de standaarddeviaties
  standev1 en standev2 volgen. De array's fracties1 en fracties2
  met de bijbehorende waarden voor de standaarddeviaties
  standev1 en standev2 worden door het programma op het scherm
  geschreven.
  Daarna worden voor beide gevallen de fracties uitgerekend
  met behulp van de bekende formule voor de Gaussische
  verdeling, gebruikmakend van standev1 en standev2.
  Deze waarden worden ook op het scherm geschreven.
  De gebruiker kan nu nagaan dat deze waarden overeen
  moeten komen met de eerder bepaalde fracties.

}
uses pam;

var
    r,n,akubus,a,i,min,muit,abol,emax:longint;
    index,index2,energie,nivnr:longint;
    gemenergiebol1,gemenergiebol2:real;
    ensom,con1,con2,standev1,standev2,fractie:real;
    fracties1,fracties2:array[1..11] of real;
    codewoord:symboolrij;

function P(x:real):real;

const u:real=0.2316419;
      b1:real=0.319381530;
      b2:real=-0.356563782;
      b3:real=1.781477937;
      b4:real=-1.821255978;
      b5:real=1.330274429;

var   v,w:real;

begin
      v:=exp(-x*x/2)/sqrt(2*pi);
      w:=1/(1+u*x);
      p:=1-v*(b1*w+b2*w*w+b3*w*w*w+b4*w*w*w*w+b5*w*w*w*w*w);
end;


begin
```

```
repeat
    writeln('geef waarden voor r en n.');
    writeln('het produkt rn mag niet groter zijn');
    writeln('dan 30');
    writeln('geef r');
    readln(r);
    writeln('geef n');
    readln(n);
until n*r<=30;
akubus:=1;
for i:=1 to r do akubus:=akubus*2;
min:=1;
for i:=1 to n do min:=min*akubus;
abol:=0;
while abol<akubus do
begin
   writeln('in te voeren waarde voor a moet');
   writeln('minimaal zijn:',akubus);
   readln(abol);
end;
init(abol,n,min,muit,emax);
gemenergiebol1:=0;
ensom:=0;
for i:=1 to abol do
begin
      fracties1[i]:=0;
end;
for index:=0 to min-1 do
begin
      encodeer(index,codewoord);
      energie:=0;
      for i:=1 to n do
      begin
           energie:=energie+codewoord[i]*codewoord[i];
           nivnr:=(codewoord[i]+1)div 2;
           fracties1[nivnr]:=fracties1[nivnr]+1;
      end;
      ensom:=ensom+energie;
end;
gemenergiebol1:=ensom/min;
standev1:=sqrt(gemenergiebol1/n);
for i:=1 to abol do fracties2[i]:=fracties1[i];
for index:=min to muit-1 do
begin
      encodeer(index,codewoord);
      energie:=0;
      for i:=1 to n do
      begin
           energie:=energie+codewoord[i]*codewoord[i];
           nivnr:=(codewoord[i]+1)div 2;
           fracties2[nivnr]:=fracties2[nivnr]+1;
      end;
      ensom:=ensom+energie;
end;
gemenergiebol2:=ensom/muit;
standev2:=sqrt(gemenergiebol2/n);
con1:=0;con2:=0;
for i:=1 to abol do
begin
      con1:=con1+fracties1[i];
      con2:=con2+fracties2[i];
end;
con1:=con1/n;
con2:=con2/n;
for i:=1 to abol do
```

```
  begin
        fracties1[i]:=(fracties1[i]/min)/n;
        fracties2[i]:=(fracties2[i]/muit)/n;
  end;
  writeln('ingevoerde r:',r);
  writeln('ingevoerde n:',n);
  writeln('ingevoerde a:',abol);
  writeln('maximale energie emax:',emax);
  writeln('eerste geval met min codewoorden:',min);
  writeln('controle1',con1);
  writeln('gem energie v.d. bol1:',gemenergiebol1);
  writeln('standaarddeviatie1:',standev1);
  writeln('fracties1:');
  for i:=1 to abol do writeln(fracties1[i]);
  writeln('tweede geval met muit codewoorden:',muit);
  writeln('controle2',con2);
  writeln('gem energie v.d. bol2:',gemenergiebol2);
  writeln('standaarddeviatie2:',standev2);
  writeln('fracties2:');
  for i:=1 to abol do writeln(fracties2[i]);
  writeln('de fracties berekend met de gausse functie');
  writeln('voor het eerste geval:');
  for a:=1 to abol do
  begin
        i:=2*a-1;
        fractie:=2*p((i+1)/standev1)-2*p((i-1)/standev1);
        writeln(fractie);
  end;
  writeln('de fracties berekend met de gausse functie');
  writeln('voor het tweede geval:');
  for a:=1 to abol do
  begin
        i:=2*a-1;
        fractie:=2*p((i+1)/standev2)-2*p((i-1)/standev2);
        writeln(fractie);
  end;
end.
```

Resultaten programma Gaussisch in de file gausuit.

```
geef waarden voor r en n.
het produkt rn mag niet groter zijn
dan 30
geef r
geef n
in te voeren waarde voor a moet
minimaal zijn:4
ingevoerde r:2
ingevoerde n:8
ingevoerde a:6
maximale energie emax:184
eerste geval met min codewoorden:65536
controle1 6.5536000000E+04
gem energie v.d. bol1: 1.4820129395E+02
standaarddeviatie1: 4.3040866329E+00
fracties1:
 3.3408927917E-01
 2.8750038147E-01
 2.1151351929E-01
 1.1175918579E-01
 4.5803070068E-02
 9.3345642090E-03
```

```
tweede geval met muit codewoorden:77056
controle2 7.7056000000E+04
gem energie v.d. bol2: 1.5019850498E+02
standaarddeviatie2: 4.3329912443E+00
fracties2:
 3.3279692691E-01
 2.8637614203E-01
 2.0660299003E-01
 1.1803104236E-01
 4.7082641196E-02
 9.1102574751E-03
de fracties berekend met de gausse functie
voor het eerste geval:
 3.5783551729E-01
 2.8945644519E-01
 1.8939756843E-01
 1.0024072805E-01
 4.2910681164E-02
 1.4856271231E-02
de fracties berekend met de gausse functie
voor het tweede geval:
 3.5561377255E-01
 2.8845713085E-01
 1.8979232248E-01
 1.0128817939E-01
 4.3842530518E-02
 1.5390835786E-02
```