Note 34220. Fejlkorrigerende kodning

Jørn Justesen, Knud J. Larsen, Erik Paaske

Juni 2021

Denne note er en lettere opdateret og moderniseret udgave af artiklen "Fejlkorrigerende kodning"af Jørn Justesen, Knud J. Larsen og Erik Paaske publiceret i magasinet Teleteknik i 1992 (side 54 - 67). Figurerne er kopieret fra den oprindelige artikel. Opdateringen er udført af Knud J. Larsen med stor hjælp fra Henrik Enggaard Hansen.

1 Introduktion

I denne artikel vil vi give en introduktion til anvendelse af fejlkorrigerende kodning i kommunikationssystemer. En meget væsentlig årsag til digitalisering af kommunikation er netop muligheden for fejlkorrektion, og fejlkorrigerende kodning er et uundværligt led i digital kommunikation. Dette skyldes to forhold: den ønskede pålidelighed kan måske ikke opnås på anden måde i et givet kommunikationssystem, eller også nås målet billigere ved anvendelse af fejlkorrigerende kodning end f.eks. gennem anvendelse af komplicerede modulationssystemer eller større effekt i transmissionen. Eksempler på det sidste er satellitkommunikation som INMARSAT og digital mobilkommunikation som GSM-systemet, 4G-LTE eller 5G, samt i optisk kommunikation, fx til 100 og 400 Gbit/s.

Hovedvægten i artiklen er lagt på strukturer i fejlkorrigerende koder og implementering af algoritmer for kodning og dekodning. Fremstillingen følger i nogen grad Litt. nr. 1, der er en meget ingeniørmæssig bog om anvendelse af fejlkorrigerende kodning. I Litt. nr. 1 og 3 kan man se forskellige kurver og andre overvejelser om samvirke mellem kodning og den øvrige del af et digitalt kommunikationssystem.

Teorien om fejlkorrigerende kodning kræver en ganske dyb matematisk indsigt i lineær algebra. Denne note forudsætter dog kun kendskab til opstilling og løsning af lineære ligninger med matricer og matrixregning. Vi præsenterer nogle begreber og metoder, som kompletterer dette kendskab og udgør forudsætningerne for de mere komplicerede koder. Med hensyn til matematik følger fremstillingen Litt. nr. 4. Eksemplerne i artiklen er udvalgt, således at læseren selv kan skaffe sig lidt føling med metoderne, f.eks. ved at udarbejde mindre PC programmer.

Fejlkorrigerende koder er et aktivt forskningsområde på DTU Fotonik, og forfatterne har deltaget i flere projekter vedrørende kodernes praktiske anvendelser, bl.a. i forbindelse med projekter, der er finansieret af ESA.

2 Støj og kodning

Det er almindeligt kendt, at overførsel af digitale informationer over en transmissionskanal sædvanligvis bevirker fejl i de overførte informationer. Under ét kalder vi dette støj. Støj er en stokastisk proces og kan derfor ikke beskrives deterministisk. Vi vil ikke her gå nærmere ind på beskrivelse af støj, men berøre emnet fra tid til anden fra forskellige vinkler.

Tidligere troede man, at støj tilføjede informationsoverførelsen en uoprettelig skade, men efter indførelse af begrebet kodning er det blevet klart, at det også på støjinficerede transmissionskanaler er muligt at overføre data med vilkårlig god pålidelighed, blot man overholder visse spilleregler som beskrevet i kommunikationsteorien (Litt. nr. 3).

Før vi præciserer udtrykket kodning nøjere, vil vi se på visse forhold om fejl i sekvenser. Ser man på blot ét overført symbol, er det vanskeligt at sige, om der var fejl på netop dette. selv om man kender sandsynligheden for fejl på et symbol. Ser man derimod på en sekvens af symboler, kan man give mere præcise udsagn f.eks. om sandsynligheden for mere end et antal fejl blandt disse, og når meget lange sekvenser anvendes, bevirker sandsynlighedsregningens centrale grænseværdisætning (se f.eks. Litt. nr. 3), at spredningen i fejlantallet aftager. Dette gør, at opgaven med at korrigere fejl bliver mere veldefineret for (lange) sekvenser af symboler.

Når man har denne opgave nogenlunde veldefineret, kan man beskæftige sig med at gøre de transmitterede sekvenser så forskellige som muligt, således at man. selv om det veldefinerede antal fejl indtræffer, er i stand til at finde den mest sandsynlige afsendte sekvens og derigennem afgøre, hvilken information, der mest sandsynligt er afsendt. Hvis der skal transmitteres K symboler (informationssymboler) hver valgt ud af et alfabet med Q symboler og lige sandsynlige, er der ingen mulighed for at sige, hvilken sekvens der blev transmitteret, når man har modtaget K symboler med nogle ukendte fejl (både fejlplaceringer og fejlværdier er ukendte). Hvis man i stedet for K symboler sender N symboler (kodesymboler), hvor $N > K$, er der mulighed for at vælge Q^K af disse sekvenser, så de bliver forskellige på mindst D pladser. Derved bliver det muligt at vælge den mest sandsynlige afsendte sekvens, så længe antallet af fejl er mindre end $D/2$.

Oversættelsen fra informationssymboler til kodesymboler:

$$C : \mathbf{x} = (x_1, x_2, \ldots x_K) \rightarrow C(\mathbf{x}) = (c_1, c_2, \ldots c_N)$$

kaldes *kodning*. De c-sekvenser, der kan blive resultatet af oversættelsen. kaldes en *(fejlkorrigerende) kode*. eller hvis man skal udtrykke sig kvantitativt, en *(N, K)-blokkode*. Ordet "blokkode" beskriver, at blokke af informations-symboler kodes til blokke af kodesymboler uden nogen form for afhængighed mellem de enkelte blokke. I et senere afsnit præsenterer vi kodning. hvor der tillige indføres afhængighed mellem blokkene. De c-sekvenser, der indgår i koden, kaldes kodeord. Processen at finde den mest sandsynlige x -sekvens ud fra en modtaget sekvens med fejl kaldes dekodning. Da der er flere mulige c-sekvenser end x-sekvenser,

(0,0,0,0,0,0,0,0)
(0,0,0,1,0,1,1,1)
(0,0,1,0,1,0,1,1)
(0,0,1,1,1,1,0,0)
(0,1,0,0,1,1,0,1)
(0,1,0,1,1,0,1,0)
(0,1,1,0,0,1,1,0)
(0,1,1,1,0,0,0,1)
(1,0,0,0,1,1,1,0)
(1,0,0,1,1,0,0,1)
(1,0,1,0,0,1,0,1)
(1,0,1,1,0,0,1,0)
(1,1,0,0,0,0,1,1)
(1,1,0,1,0,1,0,0)
(1,1,1,0,1,0,0,0)
(1,1,1,1,1,1,1,1)

Figur 1: Eksempel på (8,4) blokkode.

har vi altså tilføjet noget overflødigt, redundans, til informationssekvensen. Dette skal ikke nødvendigvis opfattes som om, at der er tilføjet nogle symboler, men det udtrykker blot at $Q^N > Q^K$. Vi har stiltiende forudsat, at alfabetet for x og c er det samme, men redundans kan faktisk også tilføjes ved, at $N = K$, mens antallet af mulige kodesymboler er $> Q$. Denne situation findes i systemer, hvor kodning og modulation er integreret. For at få et enkelt udtryk for redundansens størrelse taler man om kodens hastighed. som defineres ved udtrykket

$$\mathbf{R = K/N}$$

forudsat at informations- og kodealfabet er identiske. Det er klart ud fra ovenstående betragtninger vedrørende fejlantallet, at lange koder (d.v.s. store N) nok opfører sig mest regelmæssigt og er nemmere at analysere, og det er da også sådan, at meget små fejlsandsynligheder i beslutningen får man kun ved tilsvarende lange koder. Implementering af disse støder imidlertid på vanskeligheder, både hvad angår det praktiske omfang og den lange forsinkelse, beslutninger kan afgives med. Vi vil derfor begrænse os til rimeligt korte koder.

Det følgende eksempel er valgt således, at det er overkommeligt at opskrive alle kodeord. og det kan tjene til at illustrere nogle af overvejelserne om pålideligheden af de dekodede data.

I fig. 1 er anført alle 16 kodeord i en binær ($Q = 2$) (8,4) blokkode. Ud over 0-ordet og 1-ordet består koden af 14 ord, der hver indeholder fire 0-er og fire 1-er. Alle kodeord er forskellige på mindst 4 pladser. Vi skal senere begrunde, at man kan rette de samme fejl uanset hvilket ord, der er sendt, og vi kan derfor se på det mest overskuelige tilfælde, hvor det sendte ord er 0-ordet.

Fejlene på de modtagne symboler antages at være indbyrdes uafhængige med sandsynligheden 0,01 og symmetriske, hvilket vil sige, at fejl fra 0 til 1 er lige så hyppige som fejl fra 1 til 0. Vi antager endvidere, at de sendte symboler er indbyrdes uafhængige og lige sandsynlige. Hvis det modtagne ord er 0-ordet, vil vi selvfølgelig gå ud fra, at det er sendt. Indeholder det modtagne ord netop en 1-er, er 0-ordet ligeledes mest sandsynligt. Hvis det modtagne ord derimod er (1,1,0,0,0,0,0,0), er der flere lige sandsynlige sendte ord, idet man på fire måder kan opnå det modtagne ved at indføre to fejl i et kodeord. Vi kunne så vælge et tilfældigt af disse ord, men det mest rimelige er at slette det, d.v.s. at markere, at ordet ikke kan dekodes.

Den viste kode kan altså korrigere en enkelt fejl. Sandsynligheden for, at dekodning giver et korrekt resultat kan i dette tilfælde udregnes:

$$P(\text{kor}) = 0{,}99^8 + 8 \cdot 0{,}99^7 \cdot 0{,}01 = 0{,}9973$$

d.v.s. med kodning er fejlsandsynligheden (inklusive sandsynligheden for at ordet slettes) reduceret til ca. 1/3. Dette er et eksempel på pålidelighedsforbedring ud fra givne forudsætninger (f.eks. signal/støjforhold (SNR) i transmissionskanalen). Man kunne også ønske at reducere kravene til transmissionskanalen for opnåelse af samme fejlsandsynlighed, 0,01. Regnes dette ud som i ovenstående formel, ses en fejlsandsynlighed i transmissionen på 0,0189 at være tilstrækkelig. Ved et almindeligt modulationssystem som PSK svarer en sådan forøgelse til en reduktion i det nødvendige SNR på 1 dB. Imidlertid skal signalenergien nu anvendes på 2 kanalsymboler for at få overført 1 informationssymbol. Dette kræver 3 dB mere, så netto giver kodningen et tab på 2 dB. Enhver kode giver et tab for dårlige SNR, men de længere koder giver en kodningsgevinst regnet ud på samme måde for de lidt bedre SNR. Den viste kode er for kort til nogensinde at give en gevinst. I et senere afsnit ser vi på, hvordan tabet kan indvindes igen. Kodningsgevinsten skal angives ved en given fejlsandsynlighed (her 0,01) og anvendes ofte som mål for forbedringen opnået ved fejlkorrigerende kodning. Denne slags udregninger er selvfølgelig kun relevante, hvor det er muligt at opnå forbedringer ved forøgelse af SNR. Ovenfor foretog vi uden videre en sammenligning af bitfejsandsynligheden i det ukodede tilfælde og blokfejsandsynligheden efter dekodningen. Denne sammenligning kunne gøres mere rimelig ved, at man bestemmer antallet af forkerte bit i de dekodede blokke. Hvis de modtagne data skal bruges som blokke, er det dog mere interessant at sammenligne med sandsynligheden for fejl i en tilsvarende blok uden kodningen. Hvis sandsynligheden for fejl i en blok på 4 bit skal vare 0,01, skal effekten øges med 1,6 dB. Hermed er tabet reduceret til 0,4 dB. Det er imidlertid i mange anvendelser en stor fordel, at de fleste blokke med mere end én fejl slettes i stedet for at blive videregivet til brugeren som data. Brugeren kan så vælge at se bort fra en måling eller interpolere mellem omgivende værdier. Ofte er det nyttigt at specificere et kodet systems egenskaber ved to størrelser: sandsynligheden for at datablokke tabes i transmissionen, $P(\text{slet})$, og sandsynligheden for udetekterede fejl, $P(\text{fejl})$. Ved anvendelse af passende lange koder kan den sidste størrelse uden store omkostninger gøres forsvindende lille. For koden i fig. 1 kan sandsynligheden

for egentlige fejl beregnes, idet denne situation optræder, når der er 3, 5 eller 7 fejl, samt i de situationer hvor fejlene svarer til et af de 15 kodeord $\neq 0$.

$$P(\text{slet}) = 0{,}00264 \text{ og } P(\text{fejl}) = 0{,}00005$$

Dekodning ved hjælp af en tabel med alle modtagne ord som indgang er en overkommelig metode, når N ikke er for stor. Hver indgang i denne tabel indeholder den mest sandsynlige sekvens, der med den givne støj kunne resultere i pågældende 8-bits sekvens. Det er imidlertid klart, at dekodning af en kode med $N > 32$ vil være umulig med denne metode.

Det er vigtigt at kunne gøre sig betragtninger om, hvor god en kode er, d.v.s. hvor forskellige er kodeordene. Et mål for dette er (Hamming-) afstanden mellem to sekvenser, der defineres som antallet af pladser på hvilke de to sekvenser adskiller sig. Det er i princippet nødvendigt at kende kodens totale afstandsstruktur, d.v.s. alle Hammingafstande mellem par af kodeord, men ofte er det sådan, at sandsynligheden for et fejlmønster aftager med antallet af fejl, så man er meget interesseret i minimumsafstanden, $D_{\min}$. Koden i fig. 1 ses at have minimumsafstanden 4. Antagelsen om afhængigheden mellem sandsynlighed og antallet af fejl holder ikke altid. F.eks. findes visse modulationssystemer, hvor fejl altid fremkommer som dobbeltfejl. Hvis alle informationssekvenser er lige sandsynlige, og sandsynligheden for et fejlmønster aftager med fejlantallet, er den bedste dekodningsmetode at knytte alle de mulige modtagne sekvenser, som ligger nærmest til et kodeord sammen med dette kodeord. En sådan metode kan i hvert fald korrigere

$$t = \lfloor (D_{\min} - 1)/2 \rfloor$$

hvor $\lfloor x \rfloor$ betyder største heltal $\leq x$. Mange dekodningsmetoder kan netop korrigere op til dette antal fejl uafhængigt af deres "retning".

3 Lineære koder

Det er temmeligt kompliceret at implementere kodning og specielt dekodning efter ovenstående opskrift. Imidlertid virker de såkaldte lineære koder lige så godt som de ovennævnte. At en kode er lineær kan udtrykkes som

$$C(\boldsymbol{x} + \boldsymbol{y}) = C(\boldsymbol{x}) + C(\boldsymbol{y}) \text{ og } C(a\boldsymbol{x}) = aC(\boldsymbol{x})$$

Elementerne i $\boldsymbol{x}$ og $\boldsymbol{y}$ er som bekendt valgt ud af et alfabet. Imidlertid har ovennævnte udtryk ikke nogen umiddelbar mening, hvis de ikke er defineret i en algebraisk struktur. Den struktur, der er nødvendig, er et såkaldt endeligt legeme, der behandles i næste afsnit. For forståelsen af dette afsnit er det tilstrækkeligt at betragte binære vektorer, d.v.s. hvor elementerne er 0 og 1. De to regneoperationer er de logiske operationer "eksklusiv-eller"(betegnet ved $+$) og "og"(betegnet ved $\cdot$). Da $1 + 1 = 0$, er $+$ og $-$ identiske for dette legeme. Alle de fremførte betragtninger gælder imidlertid for et vilkårligt endeligt legeme.

En følge af den lineære egenskab er, at D_{min} for koden kan bestemmes på en enkel måde. Det er klart, at Hamming-afstanden mellem to kodeord $\boldsymbol{a}$

og $\boldsymbol{b}$ kan findes som antallet af pladser $\neq 0$ i $\boldsymbol{a} - \boldsymbol{b}$, der kaldes (Hamming-) vægten af $\boldsymbol{a} - \boldsymbol{b}$. P.g.a. den lineære egenskab er $\boldsymbol{a} - \boldsymbol{b}$ imidlertid selv et kodeord, så D_{min} kan findes som w_{min}, der kaldes kodens minimumsvægt. For lineære koder er afstandsstrukturen altså noget nemmere at finde, selv om det for lidt længere koder er ganske besværligt. Det er således nogenlunde nemt at beregne sandsynligheden for dekodningsfejl, og som regel er det kun nødvendigt at medtage sandsynligheder for fejl svarende til kodeord med de mindste vægte.

Fra den lineære algebra ved vi, at den lineære afbildning C kan skrives som en matrixmultiplikation:

$$C(\mathbf{x}) = \mathbf{x} \cdot \mathbf{G}$$

hvor $\mathbf{G}$ er en matrix med K rækker og N søjler. $\mathbf{G}$ kaldes generatormatricen for koden C. Matricens rækker er såkaldte basisvektorer for det undervektorrum, koden udspænder. Dette betyder, at enhver matrix, der kan fremkomme af $\mathbf{G}$ ved rækkeoperationer, også kan bruges som generatormatrix for koden. De angiver godt nok en anden afbildning mellem informationssekvenser og kanalsekvenser, men kodens egenskaber er uafhængige af denne afbildning. Dette gælder også, hvis søjler ombyttes i $\mathbf{G}$. Ved anvendelse af rækkeoperationer og evt. søjleombytninger kan $\mathbf{G}$ altid bringes på følgende form:

$$\mathbf{G} = [\mathbf{E}|\mathbf{R}]$$

hvor $\mathbf{E}$ er en $K * K$-enhedsmatrix, d.v.s. den har 1 i diagonalen og 0 udenfor. Kodningen af fejlkorrigerende koder kan karakteriseres i to kategorier: systematisk og ikke-systematisk. I den første kategori forekommer informationssymbolerne x direkte i kodeordet (ikke nødvendigvis på de første pladser), og de øvrige $N - K$ symboler er redundanssymboler. Det er nemt at se, at hvis $\mathbf{G}$ har ovenstående form, bliver kodningen systematisk, og informationssymbolerne forekommer på de første K pladser. En ligning af formen

$$p_1 \cdot c_1 + p_2 \cdot c_2 + \ldots + p_N \cdot c_N = 0$$

som er opfyldt for alle kodeord i koden C, kaldes et paritetcheck for C. Det ses, at hvis alle $p_i = 1.$ bliver ovenstående opfyldt, hvis der er et lige antal 1-ere i kodeordet (lige paritet). Det er altså en generalisation af det almindeligt kendte paritetscheck.

Alle paritetscheck for en lineær kode C udgør selv en lineær kode, og kaldes generatormatricen for denne $\mathbf{H}$, fås

$$\mathbf{H} \cdot \mathbf{C}(\mathbf{x})^T = \mathbf{0} = (0,0,\ldots,0)^T \text{ for ethvert } \mathbf{x}$$

hvor ^{T} betyder transponering. Når $\mathbf{G} = [\mathbf{E}|\mathbf{R}]$, vil

$$\mathbf{H} = \left[-\mathbf{R}^T|\mathbf{E}\right]$$

give paritetscheckmatricen, idet $\mathbf{H}\cdot(\mathbf{x}\cdot\mathbf{G})^T = \left(\mathbf{H}\mathbf{G}^T\right)\cdot\mathbf{x}^T = \left(-\mathbf{R}^T + \mathbf{R}^T\right)\cdot\mathbf{x}^T = \mathbf{0}$. Foreløbigt ser det jo ikke enklere ud, men forestiller man sig et kodeord $\mathbf{C}(\mathbf{x})$

transmitteret over en transmissionskanal, hvor der indtræffer fejl på nogle af $\mathbf{C}(\mathbf{x})$'s koordinater, kan man skrive det modtagne som $\mathbf{r} = \mathbf{C}(\mathbf{x}) + \mathbf{e}$, hvor $\mathbf{e}$ er en ukendt vektor med 0 på koordinater uden fejl og fejlværdierne på koordinaterne med fejl. Man får da

$$\mathbf{H} \cdot \mathbf{r}^T = \mathbf{H} \cdot (\mathbf{C}(\mathbf{x}) + \mathbf{e})^T = \mathbf{H} \cdot \mathbf{C}(\mathbf{x})^T + \mathbf{H} \cdot \mathbf{e}^\mathbf{T} = \\ \mathbf{0} + \mathbf{H} \cdot \mathbf{e}^T = \mathbf{H} \cdot \mathbf{e}^T$$

som altså kun afhænger af fejlen, ikke af de sendte data. Dekodning er nu meget enklere: Man etablerer en sammenhæng mellem $\mathbf{s} = \mathbf{H} \cdot \mathbf{e}^T$, som kaldes syndromet, og $\mathbf{e}$. Når syndromet $\mathbf{s}$ er fundet ved multiplikation af det modtagne med $\mathbf{H}$, bruger dekoderen denne sammenhæng til at finde den ukendte $\mathbf{e}$, og man kan gendanne $\mathbf{C}(\mathbf{x})$ ved at subtrahere: $\mathbf{C}(\mathbf{x}) = \mathbf{r} - \mathbf{e} = (\mathbf{C}(\mathbf{x}) + \mathbf{e}) - \mathbf{e}$. Da syndromet kun afhænger af fejlmønstret, er dekodning for lineære koder langt mere enkel end for de mere generelle koder i første afsnit. Syndromet $\mathbf{s} = \mathbf{H} \cdot \mathbf{e}^T$ er en søjlevektor, men vi tillader os af praktiske grunde at skrive det som en rækkkevektor. Koden fra fig. 1 ses at være lineær og kodningen systematisk med

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

og

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$\mathbf{H}$ er fundet ved ovenstående fremgangsmåde, men for denne specielle kode kunne $\mathbf{G}$ være benyttet som paritetscheckmatrix også. Det er nemt at se, at en enkelt fejl giver et syndrom lig den søjle i $\mathbf{H}$, som har samme nummer som den position, hvor fejlen var. Da disse søjler er forskellige, er det nemt at rette fejl. 0 fejl viser sig selvfølgelig som (0,0,0,0). Forsøger man med 2 fejl, f.eks. på position 1 og 2 (fejlmønster (1,1,0,0,0,0,0,0)) fås (0,0,1,1), og dette fremkommer også ved 3 andre fejlmønstre: (0,0,0,0,0,0,1,1),(0,0,0,1,1,0,1,0,0) og (0,0,1,0,1,0,0,1), så man kan ikke korrigere to fejl. Den lineære betragtning af koden giver altså ganske samme resultat som i foregående afsnit, hvor der var 4 mulige kodeord for modtagelse af (1,1,0,0,0,0,0,0)

Det er nu nemt at se, hvordan man kan konstruere en kode, der kan korrigere enkeltfejl. Man kan nemlig opstille $\mathbf{H}$ med m rækker og alle $2^m - 1$ kombinationer af m bit $\neq (0,0,\ldots,0)$ som søjler. En enkeltfejl giver da et syndrom lig den pågældende søjle, og da disse er forskellige, kan fejlen korrigeres. På den måde fås altså enkeltfejlkorrigerende koder med $N = 2^m - 1$ og $K = N - m = 2^m - m - 1$, d.v.s. $(N,K) = (3,1),(7,4),(15,11)$ o.s.v. De således konstruerede koder med $D_{\min} = 3$ kaldes Hamming-koder efter R.W. Hamming, som opfandt disse verdens første koder i 1949. Koden i fig. 1 er en såkaldt udvidet Hamming-kode,

da den kan fremkomme ved at udvide ethvert kodeord i en Hamming-kode med 1 symbol valgt således, at der er et lige antal 1-er i hvert ord. De udvidede Hamming-koder har altså $N = 2^m$, $K = 2^m - m - 1$ og $D_{\min} = 4$.

I nogle tilfælde er transmissionskanalen sådan indrettet, at den kan fortælle, at den om nogle af de modtagne kanalsymboler er usikker på værdien, mens den om andre er ret sikker. F.eks. kunne man måske have målt en spænding på transmissionskanalen midt imellem værdierne svarende til 0 og 1. En sådan oplysning om et symbol kaldes en sletning for at antyde, at man er bedst tjent med at se bort fra det modtagne symbol. Generelt kan man med en fejlkorrigerende kode med minimumsafstand $D_{\min}$ finde de korrekte symboler for $D_{\min} - 1$ slettede symboler, så hvis man har adgang til denne information, kan dobbelt så mange symboler korrigeres. Dette kan indses ved at betragte $\mathbf{H} \cdot \mathbf{e^T} = \mathbf{s}$ som et ligningssystem med $\mathbf{D}_{\min} - 1$ ubekendte (på de slettede pladser i $\mathbf{e}$, som har 0 på de øvrige) og bemærke, at de tilsvarende søjler i $\mathbf{H}$ altid vil være lineært uafhængige (ellers var D_{min} ikke minimumsvægten), så ligningssystemet kan løses. Tillader man foruden s slettede symboler f ukendte fejlpositioner, kan denne kombination korrigeres, hvis

$$2 \cdot f + s < D_{\min}$$

Som man kunne se i eksemplet baseret på fig. 1 kan et syndrom svare til flere fejlmønstre, og den bedste dekoder får man ved at vælge det mest sandsynlige mønster blandt disse, som regel det med lavest vagt. Mere matematisk udtrykt skyldes de flere mulige fejlmønstre, at ligningssystemet $\mathbf{H} \cdot \mathbf{e}^T = \mathbf{s}$ har $N - K$ ligninger med N ubekendte, så det kan man ikke bare løse. Hvis man kan implementere en tabel med Q^{N-K} indgange, kan man lave en tabel over de bedste fejlmønstre som funktion af de mulige syndromer. Det vil give en meget hurtig dekodning. Ofte er dette ikke muligt, hvorfor man må have en eller anden dekodningsalgoritme til at rette fejlene. I et senere afsnit omtaler vi nogle meget vigtige eksempler på koder, hvor der findes algoritmer til at behandle et meget stort antal syndromer. Man kender mange koder. hvor effektive dekodningsalgoritmer ikke kendes. Matrixmultiplikationerne i $\mathbf{x} \cdot \mathbf{G}$ og $\mathbf{H} \cdot \mathbf{r}^T$ kan selvfølgelig udføres ved at multiplicere ("og") hver bit i vektoren med den pågældende bit i matricen og addere ("eksklusiv-eller") resultaterne sammen. I en computer er dette tidskrævende. Ofte findes de logiske operationer for f.eks. 16 bit ad gangen, og man kan gøre operationen langt hurtigere ved for hver bit i vektoren at multiplicere med en hel række i matricen ad gangen og derefter addere resultatet til det foregående resultat. Et lille program illustrerer dette:

```
C:=0;
FOR I:=1 TO K DO
IF X[I]=1 THEN C:=C XOR G[I];
{C er nu et kodeord}
```

C og G skal være datatyper med mindst N bits. og G[I] er en række i $\mathbf{G}$. Skulle så brede typer ikke findes, vinder man stadig meget ved at dele op i f.eks. 16-bits typer. Denne metode er muligvis også for langsom, så hvis tabelopslag ikke umiddelbart kan realiseres, kan man benytte den lineære egenskab som i følgende

eksempel:

$$\begin{aligned}\mathbf{x} &= (x_1, \ldots, x_8, x_9, \ldots, x_{16}) \\ &= (x_1, \ldots, x_8, 0, \ldots, 0) + (0, \ldots, 0, x_9, \ldots, x_{16}) \\ &= \mathbf{x}_L + \mathbf{x}_H\end{aligned}$$

og dermed

$$\mathbf{x} \cdot \mathbf{G} = \mathbf{x}_L \cdot \mathbf{G} + \mathbf{x}_H \cdot \mathbf{G}$$

Da hver af de to delsummer kun afhænger af 8 variable, kan de forudberegnes og tabellægges i to tabeller med $2^8 = 256$ indgange, og én addition ("eksklusiv-eller") giver det endelige kodeord, som ellers ville kræve en tabel med $2^{16} = 65536$ indgange.

Mere komplicerede koder udnytter en struktur i $\mathbf{G}$ for at reducere kompleksiteten i beregningerne. Den vigtigste struktur er de såkaldte cykliske koder. som omtales nærmere efter næste afsnit.

4 Regning i endelige legemer

Formelt set foregik regningerne i det foregående afsnit i en bestemt algebraisk struktur kaldet et endeligt legeme, hvilket er en struktur med et endeligt antal elementer, to regneregler ($+$ og $\cdot$), et neutralelement svarende til hver regel (0 og 1) og et inverst element til et element for hver regel:

$$\big(x + (-x) = 0, x \cdot x^{-1} = 1(x \neq 0)\big)$$

og diverse "almindelige"regneregler. Dette ses at ligne de reelle tal, som også er et legeme, men med uendeligt mange elementer. Det simpleste eksempel på endelige legemer er regning modulo et primtal. Et endeligt legeme med Q elementer betegnes normalt GF(Q) (fra engelsk "Galois Field"), og sådanne eksisterer for alle $Q = p^m$, hvor p er et primtal og $m > 0$ et heltal. Det simpleste legeme er GF(2) hvor $+$ svarer til "eksklusiv-eller"(da $1 + 1$ modulo $2 = 0$), og $\cdot$ svarer til "og". Som nævnt i det foregående afsnit er $+$ og $-$ identiske i GF(2), da $1 + 1 = 0$. I ethvert endeligt legeme findes et såkaldt primitive element, α, hvor alle elementer $\neq 0$ kan skrives som potenser af dette element:

$$\alpha^0, \alpha^1, \ldots, \alpha^l, \ldots \alpha^{Q-2}$$

og da $\alpha^{Q-1} = \alpha^0$ er der ikke flere relevante potenser. Potensen l kaldes *logaritmen* til elementet. Bemærk, at 0 ingen logaritme har. α^0 er neutralelementet 1 for multiplikation. Multiplikation er nu let, idet

$$\alpha^m \cdot \alpha^l = \alpha^{m+l} = \alpha^{(m+l) \text{ modulo } (Q-1)}$$

p.g.a. ovennævnte sammenhæng $\alpha^{Q-1} = \alpha^0$. Multiplikation i endelige legemer er altså helt som anvendelse af logaritmetabeller for multiplikation af reelle tal bortset fra tabellernes endelige længde. I de simple legemer $GF(p)$, hvor p er et primtal, er dette selvfølgelig ikke nødvendigt. Man kan bare anvende almindelig multiplikation og reducere resultatet modulo p.

$$
\begin{array}{lll}
0 & (0,0,0,0) \text{ for } 0\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0 & \\
1 & (0,0,0,1) \text{ for } 0\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0 & \\
\alpha & (0,0,1,0) \text{ for } 0\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0 & \\
\alpha^2 & (0,1,0,0) \text{ for } 0\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0 & \\
\alpha^3 & (1,0,0,0) \text{ for } 1\alpha^3+0\alpha^2+0\alpha^1+0\alpha^0 & \\
\alpha^4 & (0,0,1,1) \text{ for } 0\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0 & ! \\
\alpha^5 & (0,1,1,0) \text{ for } 0\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0 & \\
\alpha^6 & (1,1,0,0) \text{ for } 1\alpha^3+1\alpha^2+0\alpha^1+0\alpha^0 & \\
\alpha^7 & (1,0,1,1) \text{ for } 1\alpha^3+0\alpha^2+1\alpha^1+1\alpha^0 & ! \\
\alpha^8 & (0,1,0,1) \text{ for } 0\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0 & ! \\
\alpha^9 & (1,0,1,0) \text{ for } 1\alpha^3+0\alpha^2+1\alpha^1+0\alpha^0 & \\
\alpha^{10} & (0,1,1,1) \text{ for } 0\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0 & ! \\
\alpha^{11} & (1,1,1,0) \text{ for } 1\alpha^3+1\alpha^2+1\alpha^1+0\alpha^0 & \\
\alpha^{12} & (1,1,1,1) \text{ for } 1\alpha^3+1\alpha^2+1\alpha^1+1\alpha^0 & ! \\
\alpha^{13} & (1,1,0,1) \text{ for } 1\alpha^3+1\alpha^2+0\alpha^1+1\alpha^0 & \\
\alpha^{14} & (1,0,0,1) \text{ for } 1\alpha^3+0\alpha^2+0\alpha^1+1\alpha^0 & !
\end{array}
\tag{1}
$$

Figur 2: Det endelige legeme $GF(16)$. Med ! har vi noteret, hver gang $\alpha^4 = \alpha + 1$ er anvendt til generering af repræsentationen af et legeme.

For at behandle addition i legemer GF(p^m) med $m > 1$ bliver man desværre nødt til at indføre endnu en struktur. Der findes mindst ét m'te grads polynomium med koefficienter fra "grundlegemet" GF(p) som ikke kan spaltes i faktorer over GF(p), men som har det primitive element α som rod, når der regnes i GF(p^m). Da α er rod i dette *primitive polynomium*, giver polynomiet en sammenhæng mellem α^m og lavere potenser af α. Legemet GF(p^m) kan nu også repræsenteres ved vektorer af længde m med elementer fra GF(p) på en måde, som nemmest illustreres ved eksemplet i fig. 2. På denne figur anvendes det primitive polynomium $z^4 + z + 1$ til generering af GF(2^4). Addition defineres nu, ved at vektorerne i denne repræsentation adderes komponentvis i legemet GF(p). På fig. 2 gælder f.eks. $\alpha^2 + \alpha^4 = \alpha^{10}$, og huskes multiplikationen fra før, gælder f.eks. $(0,1,0,1) \cdot (1,1,1,0) = (0,0,1,1)$. Vektorerne med m elementer udgør et vektorrum, så der kan vælges andre baser for dette til repræsentation af $\alpha^0, \alpha^1, \ldots, \alpha^{m-1}$, og det primitive polynomium kan anvendes til udregning af repræsentationen for enhver højere potens af α ud fra repræsentationen af de foregående m potenser. Vi har i fig. 2 valgt den simpleste form for basis. I det følgende regner vi, når andet ikke eksplicit skrives, altid i GF(2), d.v.s. på almindelig bit. Et andet, teknisk interessant legeme er GF(256) = GF $\left(2^8\right)$ d.v.s. med bytes som elementer. Der findes tabeller over de primitive polynomier i f.eks. litt. nr. 2.

5 Cykliske koder, BCH og Reed-Solomon koder

Næsten alle lidt længere blokkoder, der kan implementeres, er cykliske. Hvad dette begreb betyder, vil vi illustrere med et eksempel. Hamming (7,4)-koden over GF(2) kan have generatormatricen

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Svarende til informationssekvenserne (1,0,1,1), (1,1,1,0) og (0,1,1,1) fås kodeordene (1,0,0,0,1,0,1), (1,1,0,0,1,1,0) og (0,1,1,0,0,0,1). D.v.s. samtlige "cykliske skift"af mønstret 1011 findes, hvor cyklisk skift betyder, at symboler, der falder ud over enden, indsættes i begyndelsen. Da ethvert kodeord kan skrives som en sum af nogle af de i alt 7 beskrevne kodeord, gælder det om ethvert kodeord, at et cyklisk skift af dette også er et kodeord. En sådan kode kaldes *cyklisk*.

Det er nemt at indse, at paritetscheckene for en cyklisk kode også må være en cyklisk kode, og at en generatormatrix for disse svarende til (7,4)-koden er

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Dette ses at være paritetscheckmatricen for en Hamming-kode som påstået, så med en bestemt ordning af symbolerne er Hamming-koder cykliske koder. Som nævnt kan generatormatricen for en cyklisk kode være K cykliske skift af et mønster med $N-K+1$ symboler. Dette mønster kan opfattes som koefficienter til et polynomium, kaldet *generatorpolynomiet*, normalt skrevet

$$G(z) = g_{N-K} z^{N-K} + \ldots g_1 z + g_0$$

Skrives informationssekvensen $\boldsymbol{x}$ på tilsvarende måde som et $K-1$'te grads polynomium $X(z)$ og kodeordet som et $N-1$'te grads polynomium

$$C(z) = c_{N-1} z^{N-1} + \ldots c_1 z + c_0$$

kan kodningen (*ikke-systematisk*) skrives som

$$C(z) = G(z) \cdot X(z)$$

Bemærk, at nummereringen af symboler er anderledes end i de foregående afsnit: Kodeordet er her $(c_{N-1}, \ldots, c_1, c_0)$. Af ovenstående udtryk ses, at det er karakteristisk, at alle kodeord har G(z) som en faktor. Dette kan også opnås ved at danne $C(z) = X(z) \cdot z^{N \cdot K} + \big(-X(z) \cdot z^{N-K} \text{ modulo } G(z)\big)$ der ved division med $G(z)$ giver rest 0. Herved fås en systematisk kodning af en cyklisk kode med generatorpolynomiet $G(z)$

Ikke alle polynomier kan anvendes som generatorpolynomier for cykliske koder: Det er en nødvendig og tilstrækkelig betingelse, at $G(z)$ går op i $z^N - 1$.

I eksemplet er $G(z) = z^3 + z + 1$, som går op i $z^7 - 1$. Der eksisterer en særlig simpel realisation af en koder ud fra dette polynomium, idet et skifteregister og nogle "eksklusiv-eller-kredse (for GF(2)) anvendes. Detaljer kan ses i litt. nr. 2. Den simple hardware realisation er årsagen til den hyppige anvendelse af CRC (engelsk "Cyclic Redundancy Check"), som f.eks. CRC-CCITT med $G(z) = z^{16} + z^{12} + z^4 + 1$ (CCITT V.41). Når implementeringen sker i en computer, er der ingen grund til at anvende skifteregistermetoden; man kan forudberegne resten for f.eks. 8 bit ad gangen og placere disse rester i en tabel.

Udregning af et syndrom er lige så simpel, idet man kan anvende resten ved division af det modtagne med $G(z)$. Når denne rest er 0, antager vi, at der ingen fejl er. Når resten er forskellig fra 0, kan den anvendes som de tidligere betragtede syndromer. Ønsker vi at konstruere en kode, der kan rette mere end én fejl, kan det ses, at den nok ikke bliver særlig interessant med $N = 7$, da vi brugte 3 redundanssymboler bare for at rette 1 fejl. Så derfor ser vi lidt på den næste Hamming-kode med $N = 15$ og $K = 11$. Et generatorpolynomium for den er $G(z) = z^4 + z + 1$, som har α som rod i GF(16), jf. eksemplet i fig. 2. Foruden α er α^2, α^4 og α^8 de andre rødder i $G(z)$. Da $C(z) = G(z) \cdot X(z)$ er α også en rod i polynomiet $C(z)$, når der regnes i GF(16). Hamming (15,11)-koden kan altså beskrives som alle de polynomier $C(z)$ af grad (højst) 14, hvorom det gælder, at

$$C(\alpha) = 0$$

eller udregnet

$$\sum_{i=0}^{14} c_i \alpha^i = 0$$

Dette er jo et paritetscheck for koden, så man kan skrive

$$\mathbf{H} = \left[\begin{array}{ccccccc} \alpha^{14} & \alpha^{13} & \alpha^{12} & \ldots & \alpha^2 & \alpha^1 & \alpha^0 \end{array} \right]$$

og anvendes vektorrepræsentationen i fig. 2 som søjler i $\mathbf{H}$, er alle disse søjler netop forskellige, som de skal være for en Hamming-kode.

Ovenstående eksempel beskriver, hvordan enkeltfejlkorrigerende koder findes ud fra en rod i $C(z)$. Man kan nu tænke sig, at flerfejlkorrigerende koder kan konstrueres ved at have nogle flere rødder i $C(z)$. Det er også tilfældet, idet konstruktionen kan generaliseres til de såkaldte *t-fejlkorrigerende BCH-koder* med $N = 2^m - 1$:

$$C\left(\alpha^j\right) = 0 \text{ for } j = 1,2,3,\ldots.2t$$

Generatorpolynomiet $G(z)$ for disse skal altså findes som det polynomium med mindst grad, hvori $\alpha^1, \alpha^2, \ldots, \alpha^{2t}$ er rødder. Det er ikke så nemt at finde $G(z)$ uden at indføre mere matematik, men litt. nr. 1-4 beskriver alle detaljer. Vi nøjes med et eksempel på en 3-fejlkorrigerende kode: $G(z) = z^{10}+z^8+z^5+z^4+z^2+z+1$ har bl.a. rødderne $\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5$ og α^6 i GF(16), men ikke α^7, så med dette generatorpolynomium fås en (15,5)-kode med $D_{min} = 2 \cdot 3 + 1 = 7$. BCH-koder kan dekodes med den algoritme, der beskrives i fig. 3 for Reed-Solomon koder. De blev fundet i $1959-60$ af Bose, Chaudhuri og Hocquenghem. I perioden $1960-75$

1. Kaldes de modtagne symboler r_i med $0 \leq i \leq N-1$, beregnes syndromer som
$$s_j = \sum_{i=0}^{N-1} r_i \alpha^{ij}, \quad j = 1,2,\ldots,2t$$

2. Vi lader s_j være koefficienter i et polynomium $S(z) = s_1 + s_2 z + \ldots s_{2t} z^{2t-1}$, som skal anvendes i den følgende del af algoritmen og sætter
$$T_0(z) = 0, R_0(z) = z^{2t}$$
$$T_1(z) = 1, R_1(z) = S(z)$$

3. Gentag herefter følgende operationer (Euklids algoritme) for $i \geq 2$ indtil graden af $R_i(z)$ falder under t:
   (a) Find resten $R_i(z)$ og kvotienten $Q_i(z)$ ved en division af $R_{i-2}(z)$ med $R_{i-1}(z)$.
   (b) Udregn $T_i(z) = T_{i-2}(z) - Q_i(z)T_{i-1}(z)$.

4. Sæt $\sigma(z) = T_i(z)$ og $\omega(z) = R_i(z)$.

5. Find rødderne i $\sigma(z)$ og gør følgende for hver rod $z = \alpha^{-j}$:
   (a) Udregn fejlen af position j som $e_j = \omega(\alpha^{-j})/\sigma'(\alpha^{-j})$ og korriger værdierne r_j til $r_j - e_j$. $\sigma'(z)$ er den differentierede af $\sigma(z)$.

Figur 3: Dekodningsalgoritme for Reed-Solomon kode.

udvikledes dekodningsalgoritmen. Denne algoritme er ret effektiv, så BCH-koder er de bedste binære koder med relevante længder og minimumsafstande.

Hvis man har binære kanalsymboler, kan man dele dem i m-bits grupper, som på samme måde som i fig. 2 kan betragtes som elementer i GF(2^m). Vi kalder igen kodeordet for $C(z)$, men nu kan koefficienterne antage værdier i GF(2^m). Hvis vi stadig kræver
$$C\left(\alpha^j\right) = 0 \text{ for } j = 1,2,3,\ldots,2t$$
fås nogle ikke-binære BCH-koder. Disse kaldes normalt *Reed-Solomon-koder* eller blot *RS-koder*, og de blev fundet af Reed og Solomon i 1960. Koderne findes selvfølgelig også for GF(p^m), men kun GF(2^m) er teknisk interessante, specielt for $m = 8$, som anvendes på CD-plader og i satellitkommunikation. Da vi nu regner i samme legeme med hensyn til rødder og koefficienter, er det nemt at finde generatorpolynomiet for en t-fejlkorrigerende RS-kode:
$$G(z) = (z-\alpha)(z-\alpha^2)\ldots(z-\alpha^{2t})$$
Kodens parametre er altså $N = 2^m - 1$ og $K = N - 2t$ med $D_{min} = 2t+1$. I fig. 3

beskrives en dekodningsalgoritme for RS-koder, som med mindre modifikationer også kan anvendes for BCH-koder.

Vi vil nu give et eksempel på anvendelse af algoritmen: Hvis vi regner i legemet GF(16), der blev vist på fig. 2, kan vi konstruere en (15,9) RS-kode med afstand 7, og det vil altså være muligt at rette 3 fejl. Antag at fejlene er på position 0, 5 og 10 og lad for nemheds skyld fejlværdierne være ens, α. Først udregnes de 6 syndromer, som jo p.g.a. lineariteten er uafhængige af det sendte kodeord, så det modtagne ord kunne være $(0,0,0,0,\alpha,0,0,0,0,\alpha,0,0,0,0,\alpha)$
$s_1 = 0, s_2 = 0, s_3 = \alpha$
$s_4 = 0, s_5 = 0, s_6 = \alpha$
Vi starter så Euklids algoritme med $R_0(z) = z^6, R_1(z) = \alpha z^5 + \alpha z^2, T_0(z) = 0$ og $T_1(z) = 1$. I første skridt findes

$$R_2(z) = z^3, T_2 = \alpha^{14} z$$

1 næste skridt findes

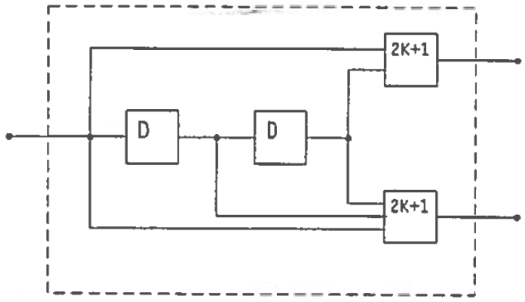
$$R_3(z) = \alpha z^2, T_3 = 1 + (\alpha z^2) \cdot (\alpha^{14} z) = 1 + z^3$$

Hermed er $\sigma(z) = 1 + z^3$ og $\omega(z) = \alpha z^2$. Den afledede af σ beregnes på sædvanlig måde, idet man dog benytter, at 3 (mod 2) er $1 : \sigma'(z) = z^2$. Det ses, at $\sigma(z)$ har rødderne $\alpha^0 = \alpha^{-15}, \alpha^5 = \alpha^{-10}$ og $\alpha^{10} = \alpha^{-5}$, hvilke svarer til de rigtige fejlpositioner. Værdien af fejlene er i alle tilfælde $\omega(z)/\sigma'(z) = \alpha \cdot z^2/z^2 = \alpha$
Algoritmen i fig. 3 er ret effektiv, idet der behandles m bits ad gangen i alle operationer.

6 Tilpasning af koder til praksis

Det er selvfølgelig temmeligt sjældent, at en naturlig opdeling af data svarer til det antal informationssymboler, der er i de beskrevne standardkoder. F.eks. genereres måske kun 22 6-bits symboler pr. naturlig datablok, hvor en RS-kode over GF(2^6) for korrektion af 8 fejl kan klare $63 - 2 \cdot 8 = 47$ symboler. Dette problem løses imidlertid let ved at anvende *afkortning*, hvilket vil sige, at et antal informationssymboler sættes lig 0, og derfor ikke behøver at blive transmitteret eller indgå i syndromberegningen. Kodens minimumsafstand ændres ikke af den grund, så i eksemplet vil (38,22)-koden kunne korrigere 8 symbolfejl ganske som (63,47)-koden kunne det.

Når man i øvrigt skal vælge koder til et bestemt formål, er der for de binære koder tabeller til rådighed, hvori de optimale koder med givne parametre N og K er beskrevet. I litt. nr. 5 findes tabeller over de minimumsafstande, der kan opnås for $1 \leq N \leq 127$ og alle $K < N$. CRC-checket, beskrevet tidligere, er i øvrigt et eksempel på afkortning af en binær kode med $N = 32767$ og $D_{min} = 4$, når den anvendes for $K < 32752$. For ikke binære koder kendes stort set kun RS-koderne.



Figur 4: Eksempel på koder for foldningskode.

7 Foldningskoder

En koder for en lineær blokkode er et lineært kombinatorisk netværk, der som tidligere beskrevet karakteriseres ved en $K \times N$ matrix $\mathbf{G}$, idet koderens uddata fås af $\mathbf{c} = \mathbf{x} \cdot \mathbf{G}$

Det er nærliggende at interessere sig for, hvad der sker, når man realiserer en koder, hvor man erstatter det kombinatoriske netværk med et sekvensnetværk. Generelt kan vi udtrykke uddata til tiden $t, \mathbf{c}_t$ som en lineær funktion f af inddata $\mathbf{x}_t$ og netværkets tilstand. $\mathbf{s}_t$, d.v.s

$$\mathbf{c}_t = f\left(\mathbf{x}_t, \mathbf{s}_t\right)$$

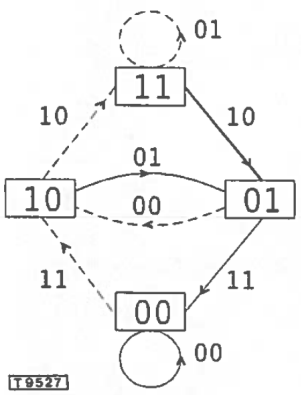
men normalt vil man realisere koderne med K skifteregistre uden tilbagekobling og med maksimal længde M. Det medfører, at vi kan skrive $\mathbf{c}_1$ på formen

$$\mathbf{c}_t = \mathbf{x}_t \cdot \mathbf{G}_0 + \mathbf{x}_{t-1} \cdot \mathbf{G}_1 + \ldots + \mathbf{x}_{t-M} \cdot \mathbf{G}_M$$

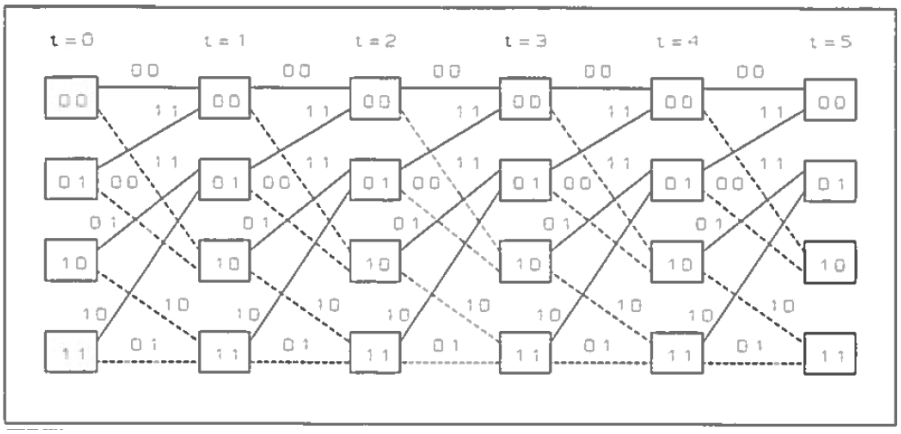
hvor de $M+1$ $K \times N$ matricer $\mathbf{G}_0, \mathbf{G}_1, \ldots, \mathbf{G}_M$ nu er generatormatricer for vores kode, og de beskriver koderen for koden. En sådan kode kaldes en *foldningskode*, fordi ligningen netop udtrykker en foldning. Et simpelt eksempel er vist i fig. 4. hvor vi har $K = 1, N = 2, M = 2$ samt $\mathbf{G}_0 = [11], \mathbf{G}_1 = [01]$ og $\mathbf{G}_2 = [11]$ Da koderen er en sekvensmaskine, kan vi beskrive den ved et tilstandsdiagram som vist i fig. 5 for vores koder i fig. 4. Fra hver af de $2^M = 4$ tilstande udgår der 2^K $(= 2)$ kanter (overgange) til de tilstande, koderen kan bringes i af de 2^K forskellige inddata. Hver kant er mærket med $\mathbf{c}_t = f\left(\mathbf{x}_t, \mathbf{s}_t\right)$, og i fig. 5 er punkterede kanter genereret af $\mathbf{x}_t = [1]$ og fuldt optrukne af $\mathbf{x}_t = [0]$

Enhver mulig kodet sekvens (kodeord) kan findes som en sekvens af kanter i tilstandsdiagrammet. Dette illustreres tydeligere, hvis vi indfører tidsdimensionen i tilstandsdiagrammet, således at vi kopierer tilstande og kanter for hver tidsenhed, hvorved vi får en gitterstruktur som vist i fig. 6. En kodet sekvens $\mathbf{c}$ er altså en vej gennem gitteret.

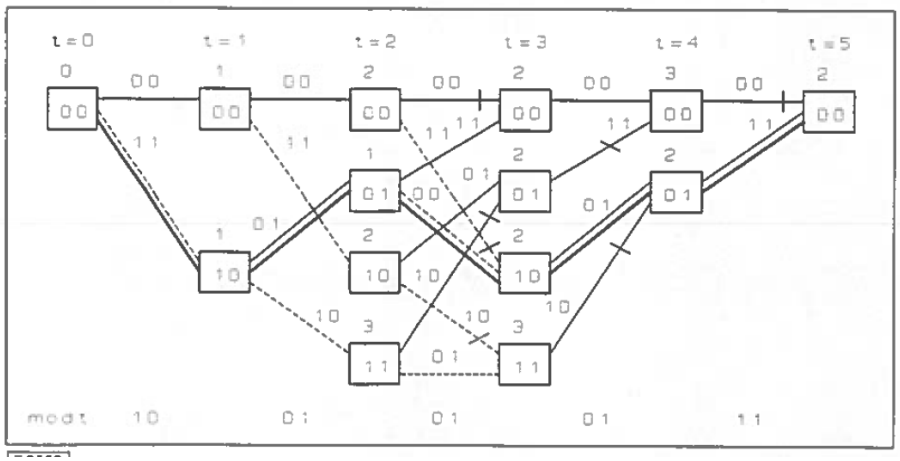
Antag nu, at vi starter med koderen i nultilstanden og koder en sekvens, som er afsluttet med 2 0-er, hvorved koderen igen når nultilstanden. Herved reduceres gitteret som vist i fig. 7. Den kodede sekvens sendes over kanalen



Figur 5: Tilstandsdiagram for koder i fig. 4.



Figur 6: Gitterstruktur for foldningskode i fig. 4.



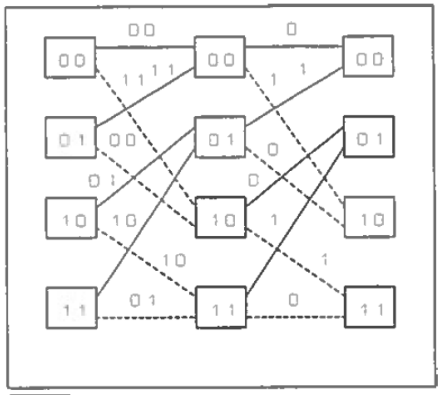
Figur 7: Gitterstruktur for koder, der starter og ender i nultilstand.

og modtages som sekvensen vist nederst på figuren. En optimal dekoder finder den vej i gitteret, som afviger fra den modtagne på det mindste antal pladser. Metoden, som kaldes *Viterbidekodning*, kan beskrives således:

Til $t = 1$ kan koderen være i en af to tilstande, og vejen til hver af disse afviger på 1 plads, hvilket er anført over tilstandskasserne. Til $t = 2$ kan koderen være i en af fire tilstande. Det antal pladser, hvor en vej til en tilstandskasse afviger fra den modtagne sekvens kaldes vejens akkumulerede metrik, og metrikværdierne påføres over tilstandskasserne. Til $t = 3$ kan koderen fortsat være i en af fire tilstande, men til hver tilstand fører to veje. Da vi skal minimere metrikken, kan vi allerede nu forkaste den af de to veje, som har størst metrik (Vist ved kryds over vejen). Den tilbageblevne vej kaldes den overlevende vej, og dens metrik påføres over tilstandskassen. Saledes fortsættes til $t = 5$, hvor vi kun har én overlevende vej med metrik 2, hvilket indikerer, at vi har rettet 2 fejl. Den overlevende vej, der er vist med fed streg, findes ved at følge vejen tilbage fra $t = 5$.

I det viste eksempel antog vi, at koderen startede og sluttede i nultilstanden. Denne antagelse er i praksis ikke nødvendig. Man kan lade koderen og dekoderen køre kontinuert og benytte sig af, at alle de 2^M overlevende veje til tidspunktet t har en fælles forhistorie for $t - L$, hvor L skal være "tilstrækkelig"stor. Ved en tilbagesøgning fra en tilfældig tilstand til tiden t afleveres kun data fra tiden for $t - L$ til brugeren. En metode er at have to tilbagesøgningsprocesser kørende simultant: den ene er startet til tiden t og den frigiver L symboler til brugeren for tiden mellem $t - L$ og $t - 2L$. Den anden starter til tiden $t + L$ og frigiver symboler til brugeren for tiden mellem t og $t - L$.

En foldningskodes fejlkorrigerende egenskaber beror på afvigelser (Hammingafstande) mellem veje i gitteret. Da koden er lineær, kan alle relevante afstande måles som Hammingafstande mellem nulsekvensen og andre veje i gitteret. Den



Figur 8: To gittersektioner for foldningskoder med punktering.

vigtigste parameter betegnes den *frie afstand*, d_{free} og defineres som den mindste vægt for en vej, der starter og ender i nultilstanden uden mellemliggende passager af nultilstanden. Koden i det viste eksempel har $d_{\text{free}} = 5$ hvilket bl.a. betyder, at den altid kan rette 2 fejl, som er omgivet af et "passende"antal fejlfri symboler. For givne værdier af K og N (f.eks. $K = 1$ og $N = 2$ som i eksemplet) kan kodens fejlkorrigerende egenskaber forbedres ved at øge ν, det totale antal af hukommelseselementer i koderen. Generelt har koderen K skifteregistre af maximal længde M, så $\nu \leq K \cdot M$. Bemærk imidlertid, at antallet af tilstande for hver sektion i gitteret bliver 2^{ν}. D.v.s. kompleksiteten af dekoderen vokser eksponentielt med ν.

Som allerede nævnt har gitteret for en (N, K) foldningskode N symboler på kanterne, og 2^K kanter udgår og ender i hver tilstand. En gitterstruktur med 2 kanter til og fra hver tilstand giver imidlertid en række fordele, når dekoderen realiseres i hardware. Det opnås med en $(N, 1)$ kode, d.v.s. en kode med $R = 1/N$, men ved hjælp af en teknik, der kaldes *punktering*, kan strukturen også opnås for nogle koder med $R = K/N, K > 1$. Punktering består i at slette en vis brøkdel af de kodede symboler inden transmissionen, så forholdet mellem informationssymboler og kodede symboler øges. Benyttes f.eks. en (2,1)-kode og slettes hvert fjerde symbol, sendes 3 symboler for hver 2 informationssymboler, d.v.s. $R = 2/3$. I et gitter for en sådan punkteret kode har hver anden gittersektion kun ét symbol på kanterne som vist i fig. 8. Da vi jo sletter symboler, er prisen for en højere hastighed en mindre d_{free}, her $d_{\text{free}} = 3$. Fordelen ved punktering er, at dekoderen for den oprindelige (2,1)-kode med minimale ændringer kan benyttes for den punkterede kode med $R = 2/3$

Den praktiske realisering af Viterbi dekodere afhænger meget af, hvor hurtigt dekodningen skal foregå. Ved lave hastigheder, kan dekodningen realiseres med op til 64 tilstande. Ved højere hastigheder opbygges dekoderen normalt af 3 moduler: 1) Et inputmodul, der ud fra de modtagne symboler beregner metrikværdier for de 2^N forskellige kanter. 2) Et ACS-modul, der for hver tilstand adderer (Add)

de akkumulerede vejmetriker til hver af de indgående kanter, sammenligner (Compare) de to indgående veje og vælger (Select) den med mindst metrik. 3) Et PSS-modul, der for hver tilstand lagrer beslutninger om den valgte vej (Path Storage) og ved en tilbagesøgningsrutine vælger (Select) den overlevende vej, der skal afleveres til brugeren. Normalt er det ACS-modulet, der sætter grænsen for, hvad der kan realiseres. Et enkelt ACS-element kan benyttes serielt af alle tilstande og er nok til at realisere en dekoder med 64 tilstande ved lavere hastigheder, medens en hurtigere dekoder kræver et ACS-element for hver tilstand. DTU udviklede i sin tid dekoderen til en kode, som anvendes af NASA og ESA, og som har 64 tilstande.

8 Bløde beslutninger

I denne introduktion til anvendelse af kodning har vi indtil nu beskæftiget os med modtagne symboler i samme alfabet som det sendte. Imidlertid vil mange modulationssystemer tillade en mere nuanceret angivelse af det modtagne, f.eks. gennem angivelse af en spænding med flere værdier end de to, der svarer til de to symboler. Denne mulighed kaldes *bløde beslutninger* i modsætning til de tidligere hårde beslutninger. I praksis skal beslutningen selvfølgelig kvantiseres, men det er karakteristisk, at det modtagne alfabet er en del større end det sendte, f.eks. 2^3 eller 2^4 symboler. I nogle situationer kan man vælge at opfatte den bløde beslutning som sammensat af det modtagne symbol og et mål for dets pålidelighed. En vigtig egenskab ved Viterbi algoritmen er muligheden for at udnytte dette uden nogen ændringer i algoritmen. Når der anvendes bløde beslutninger, er det naturligt at betegne de modtagne værdier med reelle tal i stedet for de logiske værdier 0 og 1. Vi vil vælge at lade det sendte signal svare til 1 og -1. I stedet for at anvende fejlantallet på de enkelte grene som metrik udregnes på basis af kanalens egenskaber en metrik for hvert modtaget kanalsymbol. Ofte benytter man en kvantisering af det modtagne resultat, så f.eks. 8 værdier repræsenteres ved værdierne $-7/4, -5/4, -3/4, \ldots, 5/4, 7/4$ (i praksis udføres beregningerne naturligvis med hele tal). Dekodning med bløde beslutninger giver som regel væsentlig bedre resultater end de hårde beslutninger. Udtrykt ved kodningsgevinsten er forbedringen typisk 2 dB (ved samme bitfejlsandsynlighed). Dette har bidraget meget til foldningskoders popularitet i satellitkommunikation. Muligheden for anvendelse af bløde beslutninger eksisterer også for korte blokkoder (Litt. nr. 1), f.eks. vil det ikke være så svært at anvende det på koden i fig. 1. Muligheden for at rette slettede symboler, som allerede har været omtalt, kan opfattes som det første trin, hvor et slettet symbol erstattes med værdien 0. Hvis kodesymbolerne opfattes som reelle tal, ser man, at kodeordene enten er hinandens modsatte eller er indbyrdes ortogonale. Dekodningen med bløde beslutninger kan her analyseres på samme måde som detektionen i en modulationsform med flere pulsformer (Litt. nr. 3). Det viser sig nu, at detektionen giver en forbedring på $0{,}5 - 2$ dB afhængigt af om blokfejlsandsynligheden sammenlignes med bitfejlsandsynligheden eller blokfejlsandsynligheden for kodningen. Det tab, vi omtalte i forbindelse med

korte koder, kan altså siges at skyldes overgangen fra bløde til hårde beslutninger.

9 Fejlbyger

Vi har ikke gjort nogen forudsætninger om, hvordan fejlene fordeler sig i kodeordene. Dette kan imidlertid være et problem. F.eks. laver Viterbi dekodning af foldningskoder ofte dekodningsfejl, hvis der modtages byger af fejlramte symboler. En *byge* er et afgrænset stykke af den modtagne symbolsekvens, hvor der er relativt mange fejl, mens der kun findes meget få spredte fejl uden for bygen. Man kan tænke sig transmissionskanalen som havende to tilstande: en god med relativt lille fejlsandsynlighed og en dårlig med stor fejlsandsynlighed. Dette kan selvfølgelig forekomme, hvis transmissionen er forstyrret af noget pulserende støj som f.eks. radarsignaler. I mobilkommunikation forekommer fejlbyger som følge af periodevise signalsvækkelser, der skyldes udbredelsesforholdene for radiobølger. Dette kan i nogen grad modvirkes ved at skifte frekvens mellem de forskellige dele af transmissionen (frequency-diversity).

I virkeligheden burde en forhåndsviden om, at fejlene forekommer i byger, lette dekodningen, da dekoderen så skal skelne mellem langt færre fejlmønstre. Der findes koder som er specielt konstrueret til korrektion af fejlbyger (Litt. nr. 2), og en af disse (en såkaldt Fire kode) anvendes i GSM mobil-kommunikation. En del af problemet er, at blot 1 fejl uden for bygerne bevirker dekodningsfejl, og koderne vil derfor næppe fungere tilfredsstillende i praksis. I litt. nr. 1 beskrives forskellige forhold og algoritmer for dekodning af foldningskoder i bygestøj. Korrektion af bygefejl vil som regel resultere i en stor forsinkelse, da antallet af fejl i den modtagne blok ellers bliver for varierende. I nogle situationer kan man foretrække at forsøge retransmission af de blokke, som rammes af byger, og det kan være nødvendigt at begrænse bloklængden for at få tilstrækkeligt mange blokke igennem. Det er i reglen vanskeligt at opstille en anvendelig model for bygefejlene, og dermed bliver det også vanskeligere at konstruere effektive koder. En meget almindelig metode til bekæmpelse af bygefejl er *spredning* (eng.: *interleaving*), så fejlene nærmest ser tilfældige ud for dekoderen. Som beskrevet ovenfor er der principielt et tab ved denne metode, men f.eks. i forbindelse med en god foldningskode og Viterbidekodning kan det være en anvendelig løsning, som specielt udmærker sig ved at være robust over for bygestrukturen. I litt. nr. 1 beskrives sådanne spredningsmetoder nøje. En anden almindelig metode til at bekæmpe bygefejl direkte ved kodning er anvendelse af koder med et større symbolalfabet f.eks. $2^8 = 256$. Disse symboler kan opfattes som 8 bit, og korrektion af 1 sådant symbol kan altså rette en masse fejl på en gang, hvis fejlene ligger i byger. Hvis der også forekommer spredte fejl, kan det være en fordel at benytte et sammensat system bestående af en kort binær kode (f.eks. en foldningskode) og en Reed-Solomon-kode. Sådanne *sammensatte koder* er ofte de mest effektive i praktiske systemer, hvor der stilles store krav til effektivitet og pålidelighed.

10 Kombinerede kodere og iterativ dekodning - Udviklingstendenser frem til idag

Teorien om fejlkorrigerende kodning er vokset frem siden 2. verdenskrig. I 50'erne og 60'erne var teorien et yndlingsemne for matematikere, fordi smukke matematiske strukturer pludselig kunne finde anvendelse, medens den praktiske anvendelse i kommunikationssystemer var ringe. Det skyldtes simpelthen, at man ikke rådede over en teknologi, der på økonomisk vis kunne implementere teorien. Fra 70'erne og 80'erne er billedet vendt. Teknologien er til rådighed, og hovedparten af de nye systemer benytter fejlkorrigerende koder. Eksempler fra dagligdagen er mobiltelefoner, digital TV, DVD-skiver og QR-koder.

Gruppen for kodning og visuel kommunikation ved DTU Fotonik har deltaget på området siden begyndelsen af 70'erne (tidligere ved Instituttet for Teleteknik), og gruppen har bidraget til forskningen og udviklingen på området dels ved udvikling af nye koder og dekodningsalgoritmer og dels ved implementering af praktiske systemer, ofte i forbindelse med systemer, der benyttes af ESA. Som et eksempel kan nævnes design af en Viterbi-dekoder for 150 Mbit/s.

I 1993 introducerede C. Berrou en ny måde at sammensætte koder på som fik navnet *turbo-koder*. I dette system genereres de kodede sekvenser ved to foldningskodere, der arbejder på den samme informationsstrøm i hver sin pseudo-tilfældige rækkefølge. Beslutninger taget af de to tilsvarende dekodere bliver så rimeligt uafhængige af hinanden. Man anvender dekodere, der minder om Viterbi dekodere med bløde beslutninger, men har den ekstra facilitet at de også producerer bløde beslutninger som udgangssignal. Da man har stræbt efter at gøre de to dekodere uafhængige af hinanden, kan man anvende disse beslutninger til at styrke beslutningerne i den anden dekoder og omvendt. Man skal dog huske at fratrække sit eget bidrag til beslutningen, før man anvender det, da der ellers opstår en tilbagekobling. Efter et antal iterationer får man et dekodningsresultat som i dårlige signal/støj-forhold er langt mere pålideligt end hvad tidligere sammensatte systemer gav. Turbo-kodning anvendes i 4G-LTE mobilkommunikation. I 5G anvendes LDPC (Low Density Parity Check) koder til fejlbeskyttelse af datastrømmen, her anvendes ligeledes iterativ dekodning.

Gruppen har implementeret et samlet dekodersystem for ESA i løbet af det første årti i 2000'erne. Dette indeholder en sammensat kode med en $N = 2$ foldningskode (punkterbar) med 64 tilstande efterfulgt af en Reed-Solomon (255,223)-kode med 8 bits symboler samt en turbo-kode, der kan konfigureres til forskellige kodningshastigheder fra 1/6 til 1/2. Systemet indeholder også diverse andre funktioner, f.eks. rammesynkronisering for at lokalisere kodeordene. Systemet er implementeret i alle ESAs jordstationer.

Optisk kommunikation klarede sig tidligere uden fejlkorrektion, da de trods alt ret beskedne hastigheder direkte muliggjorde en bitfejlsandsynlighed på 10^{-9}. Hvis forholdene var lidt mere vanskelige som f.eks. ved undersøiske kabler anvendte man en enkelt (255,223) Reed-Solomon kode med 8 bits symboler. Men i nyere tid er hastigheden og kravene steget betydeligt, f.eks. 400-800 Gbit/s med en fejlsandsynlighed på 10^{-15}. Effektive systemer med sådanne hastigheder

har en input fejlsandsynlighed på 10^{-3} - 10^{-2} fra kanalen, så der skal kraftige kodningssystemer til som ikke øger transmissionshastigheden for meget gennem redundansen. Et sådant system har gruppen udviklet og implementeret for hastigheder op til 1600 Gbit/s med 7% redundans. Her er koden en såkaldt *produktkode*, hvor informationen (af størrelsesordenen 1 Mbit) anbringes i en matrix, hvor hver række og søjle kodes med en BCH-kode. Dekodningen foregår så ved iterationer, hvor f.eks. først rækkerne dekodes, og resultatet af det overleveres til søjledekoderne o.s.v. Efter nogle iterationer opnås et pålideligt resultat. For tiden arbejdes med et system til 400 - 800 Gbit/s, hvor en lignende kode sammensættes med en udvidet Hamming-kode med længde 128. Denne Hamming-kode kan dekodes med bløde beslutninger ved en forholdsvis overkommelig implementering.

Litteratur

[1] G. C. Clark and J. B. Cain: "Error-Correcting Coding for Digital Communications", Plenum Press (1981) Third printing 1988.

[2] R. E. Blahut: "Theory and Practice of Error Control Codes", Addison-Wesley 1988.

[3] J. G. Proakis: "Digital Communications", McGraw-Hill 1989.

[4] R. J. McEliece: "The Theory of Information and Coding", Addision-Wesley 1977.

[5] T. Verhoeff: "An Updated Table of Minimum-Distance Bounds for Binary Linear Codes", IEEE Transactions of Information Theory, September 1987, side 665–680.

[6] J. Justesen and T. Høholdt: "A Course in Error-Correcting Codes", Second edition". European Mathematical Society Publishing, 2017.

[7] S. Lin D. J. Costello: "Error Control Coding: Fundamentals and Applications", Second edition". Prentice-Hall 2004.