SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

Självrättande Koder

av

Samuel Karlhager

2021 - No K48

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET, 106 91 STOCKHOLM

Självrättande Koder

Samuel Karlhager

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Rikard Bögvad

2021

SJÄLVRÄTTANDE KODER

Feldetektering och felkorrigering

Samuel Karlhager

Stockholms Universitet

Innehåll

Tackord

Ett stort tack till min fru Alexandra för den varma kärlek du visar mig, en kärlek som bara blir större med tiden. Ett stort tack riktas också till min handledare Rikard Bögvad. Ända sedan de inledande kurserna på grundnivå har det alltid varit ett stort nöje att förkovra sig i matematikens värld med din stöttning.

Sammanfattning:

För att kommunicera digitalt, såsom via datorer och mobiltelefoner, krävs att vi översätter vårt meddelande till ett språk som våra datorer förstår. Denna process kallas för kodning, respektive avkodning när det ska översättas tillbaka till en läsare. Här kan man skilja mellan linjära koder och cykliska koder, där den förstnämnda bygger på linjär algebra och matriser. Hammingkoder är en vanligt förekommande linjär kod. Cykliska koder, som exempelvis BCH- eller RS-koder, tar ytterligare ett steg och behandlar större datamängder och används mer frekvent i dagens teknik. För koder kan det ibland uppstå störningar som gör att budskapet riskerar förvrängas. Till koderna kan man då bygga in självrättande egenskaper som gör att meddelandet ändå kan levereras även om det sker en störning eller skada. Detta arbete syftar till att ge en överblick för olika självrättande koder, dess tillämpningar och för-/nackdelar.

Nyckelord: kodteori, självrättande koder, hammingkod, RS, Reed-Solomon

Introduktion

Bakgrund

Att vårt samhälle blir alltmer digitaliserat är ett faktum. Även om denna digitala tidsålder utgör en ytterst liten del av mänsklighetens historia så förringar det ej vilken betydelse den haft och fortsätter att ha, vare sig det gäller storföretag eller den enskilde individen. Till denna digitalisering är det av stor vikt att kunna skicka och ta emot information på ett effektivt sätt. Om tekniken felar och information går förlorad eller ej går att avläsa är den således till ingen nytta. Att teknik ibland kommer att fela är, dessvärre, ingenting vi kan komma ifrån. Däremot finns det olika sätt att upptäcka och korrigera felen som uppstår. Ett sätt att bemöta detta matematiskt är att vända sig till den gren av matematiken som kallas för *kodteori*.

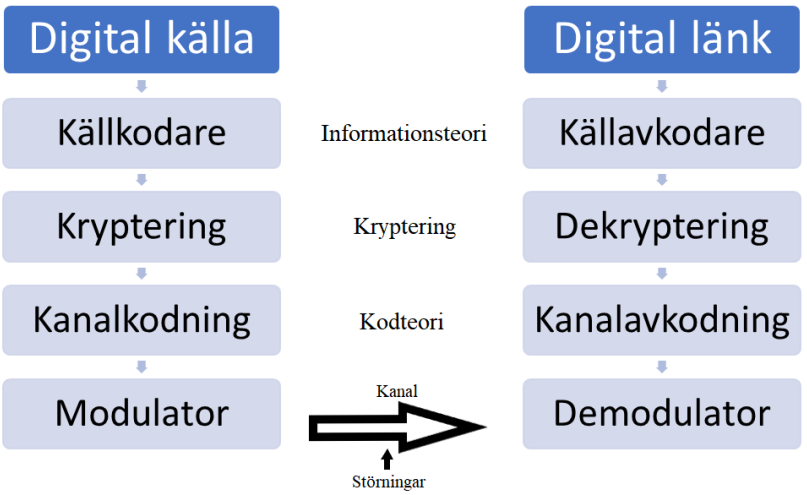
Definition från Nationalencyklopedin: “algebraisk kodteori, teori för den transformation eller kodning av information som utförs för att eliminera fel som uppstår på grund av störningar vid överföring av informationen från en sändare till en mottagare.”

Användningen av kodteori förekommer i princip på alla de ställen där vi vill överföra information från en part till en annan. Exempel på användningsområden är inom datalagring vid ex. CD-skivor och djuprymdskommunikation.[1] Till exempel kan en CD-ROM rätta till upp till 4 000 fel i rad.[2]

Ibland förekommer det faktorer som kan försvåra avläsningen av dessa, exempelvis damm på skivan eller till och med en repa. För att inte all information ska gå förlorad tillämpas ofta felrättande koder som gör att skivan ändå kan spelas upp även om det förekommer smärre fel i datalagringen. Självrättande koder har också använts vid djuprymdskommunikation, däribland i Voyager-programmet, för att komma runt problemet med att information kan skadas vid oväder och andra störningar från jordens atmosfär. [3]

Figur 1:

Beskrivning: flödesschema för olika steg i databehandling, där kodteori är en av delarna.



[1] Sarah Spence Adams, s. 5.
[2] Moreira, Farrell, s. 1.
[3] Sarah Spence Adams, s. 5.

Syfte och frågeställningar

Syftet med denna text är att redogöra för kodteori och koder som kan både upptäcka och korrigera fel som uppstår. Det kommer göras utifrån följande frågeställningar:

- Vad är en självrättande kod och vad utför den?
- Hur och när används dessa koder?
- Vilka för- och nackdelar finns med att använda självrättande koder?

Feldetektering och felkorrigering

Texten i följande del bygger på ”Discrete mathematics” av Norman L. Biggs. [4]

Vanligtvis är det sällan som vi stöter på dessa ettor och nollor vid vardagligt bruk av teknik, men datorer är i grunden uppbyggd för att hantera dessa, som i sin tur kan härledas till det binära alfabetet $\{0, 1\}$. Alltså kan meddelanden skrivas som en kombination av dessa två binära bokstäver. För ett meddelande med längd *n* finns det 2^n olika ord som kan användas och den kompletta uppsättningen av dessa ord kan beskrivas som V^n, exempelvis har vi att:

$$V^2 = \{00, 01, 10, 11\}.$$

Varje symbol (alltså etta eller nolla) i ett sådant ord brukar kallas för en *bit*. En binär kod C med längd *n* är en delmängd av V^n, där de olika elementen i C kallas för kodord.
Ett meddelande kan i teorin vara av vilken längd som helst, men helst vill man inte ha för långa meddelanden eftersom det då t.ex. tar längre tid att skicka det.

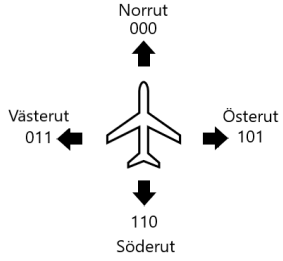
Tabellen nedan ger tre exempel på koder för att skicka meddelandena: *norrut, söderut, västerut, österut*.

Tabell 1

Kod	**Kodlängd**	**norrut**	**söderut**	**västerut**	**österut**
C_1	2	00	10	01	11
C_2	3	000	110	011	101
C_3	6	000000	111000	001110	110011

Följande figur beskriver hur texten ovan kan se ut i bild.

Figur 2:



[4] Biggs, s. 338-339.

ASCII-koder

Texten från detta stycke är baserad på http://cactus.io/tutorials/web/what-is-an-ascii-code.

ASCII är en kod som använder tal för att representera bokstäver och symboler. Förkortningen står för American Standard Code Information Interchange. Varje bokstav och symbol tillskrivs ett tal mellan 0 och 127. Detta gör man för att mikroprocessorer endast förstår bits och bytes, vilka talen kan representeras i. Vill man till exempel skriva ordet ”Hej” i ASCII-kod blir det 72 101 106, vilket i sin tur blir 01101000 01100101 01101010 i binär form. I den ursprungliga koden finns dock begränsat med symboler, vilket har medfört skapandet av utökad ASCII-kod, för att även ha med exempelvis symbolerna å, ä och ö. ASCII-koder är faktiskt äldre än internet självt och såg sin första användning vid teleprinters och mekaniska skrivare. Den ursprungliga ASCII-koden innehåller dock bara 128 tecken, i princip bara det engelska alfabetet. I detta fall är HTML-koder mer lämpligt om man arbetar i en webbläsare, för den har bättre stöd för fler tecken, men i övrigt är det ASCII-koder som är den underliggande grunden för den mesta programmeringen och kommunikationen av text som sker via teknik.

Linjära koder

Texten från detta stycke och framåt är baserat på ”Introduction to algebraic code theory”, av Sarah Spence Adams, s. 17 och vidare.

Koden *C* är en delmängd av något vektorrum V^n. I V^n kan man lägga ihop element genom att man adderar ettor och nollor på samma plats modulo 2. En kod som är sluten under addition kallas för en *linjär* kod.

Definition 1: En kod C av längd n är linjär om det är så att två kodord i *C* har en summa i C.

Nedan följer några exempel på linjära koder.

Exempel 1: Repetitionskoden {00000, 11111} av längd 5 är en linjär kod. Oavsett vilka två kodord man väljer får man ett kodord som redan ligger i mängden:
00000+00000=00000 ∈ C
00000+11111=11111 ∈ C
11111+00000=11111 ∈ C
11111+11111=00000 ∈ C

Linjära koder har stor praktisk användning av flera skäl, till exempel att de är så enkla att konstruera. Dessutom är själva kodningen av linjära koder snabb och enkel. Avkodningen brukar också underlättas av linjäriteten av en kod. Koderna som hädanefter nämns i texten kommer, om inget annat sägs, vara av binär karaktär.

Definition 2: *Hammingtyngden* $w(\mathbf{c})$ av ett kodord $\mathbf{c}$ är antalet nollskilda element i kodordet. T.ex. är $w(11000) = 2$, $w(111) = 3$ och $w(0000000) = 0$.

Definition 3: *Hammingavståndet* mellan två kodord, betecknat $\mathrm{d}(\mathbf{x}, \mathbf{y})$, är antalet platser i där kodorden skiljer sig mellan varandra. T.ex. är $\mathrm{d}(0000,1110) = 3$ och $\mathrm{d}(11,10) = 1$.

Definition 4: Det *minsta (hamming-)avståndet* för en kod C är det minsta avståndet som existerar mellan två kodord i koden: $\mathrm{d}(\mathrm{C}) = \min\{\mathrm{d}(\mathbf{x}, \mathbf{y}) | \, \mathbf{x} \neq \mathbf{y}, \mathbf{x}, \mathbf{y} \in \mathrm{C}\}$.

Exempel 2: För koden $\mathrm{C} = \{00000, 10001, 11111\}$ är det minsta avståndet 2 eftersom $\mathrm{d}(00000,10001) = 2$, $\mathrm{d}(00000,11111) = 5$ och $\mathrm{d}(10001,11111) = 3$.

Det finns också ett annat sätt att beskriva det minsta avståndet.

Från s. 17:
Sats 1: Det minsta avståndet, d(C), av en linjär kod C är samma som $\mathrm{w}^*(\mathrm{C})$, tyngden av det kodord som har minst tyngd och som inte bara består av nollor.

Bevis: Det finns kodord **x** och **y** i koden C så $\mathrm{d}(\mathrm{C}) = \mathrm{d}(\mathbf{x}, \mathbf{y})$. Av definitionen för ett hammingavstånd kan detta skrivas som $\mathrm{d}(\mathrm{C}) = \mathrm{w}(\mathbf{x} - \mathbf{y})$. Här ser vi att $\mathbf{x} - \mathbf{y}$ också är ett kodord i C på grund av dess linjäritet. Eftersom $\mathrm{w}^*(\mathrm{C})$ är tyngden av det kodord som har lägst tyngd har vi att $\mathrm{w}^*(\mathrm{C}) \leq \mathrm{w}(\mathbf{x} - \mathbf{y}) = \mathrm{d}(\mathrm{C})$. Samtidigt existerar det något kodord $\mathrm{c} \in \mathrm{C}$ så att $\mathrm{w}^*(\mathrm{C}) = \mathrm{w}(\mathrm{c})$. Enligt definitionen för vikten av ett kodord kan vi skriva $\mathrm{w}(\mathrm{c}) = \mathrm{d}(\mathrm{c}, 0)$. Eftersom såväl c som 0 är kodord måste avståndet mellan dem vara större än eller lika med det minsta avståndet i koden: $\mathrm{d}(\mathrm{c}, 0) \geq \mathrm{d}(\mathrm{C})$. Dessa två (o)likheter visar att $\mathrm{w}^*(\mathrm{C}) \geq \mathrm{d}(\mathrm{C})$.

Eftersom både $\mathrm{d}(\mathrm{C}) \geq \mathrm{w}^*(\mathrm{C})$ och $\mathrm{d}(\mathrm{C}) \leq \mathrm{w}^*(\mathrm{C})$ är det klart att $\mathrm{d}(\mathrm{C}) = \mathrm{w}^*(\mathrm{C})$.

■

Teorem 1:

1. En kod C kan upptäcka upp till s fel i ett kordord, om $\mathrm{d}(\mathrm{C}) \geq \mathrm{s} + 1$.
2. En kod C kan rätta upp till t fel i ett kordord, om $\mathrm{d}(\mathrm{C}) \geq 2\mathrm{t} + 1$.

Bevis

1. Anta $\mathrm{d}(\mathrm{C}) \geq \mathrm{s} + 1$ och att ett kodord **c** skickas samt att $\mathrm{s} \geq 1$ eller ett färre antal fel uppstår vid sändningen. Eftersom det erhållna kodordet skiljer sig från kodorden i C på åtminstone ett ställe kan det erhållna kodordet inte vara något legitimt kodord, och felen är därmed upptäckta.
2. Anta $\mathrm{d}(\mathrm{C}) \geq 2\mathrm{t} + 1$ och att vi skickar ett kodord **x** och att det erhållna kodordet, **r**, innehåller t eller ett färre antal fel, då är $\mathrm{d}(\mathbf{x}, \mathbf{r}) \leq \mathrm{t}$. Låt **x'** vilket annat kodord som helst än **x**. Då kommer $\mathrm{d}(\mathbf{x}', \mathbf{r}) \geq \mathrm{t} + 1$, för annars blir $\mathrm{d}(\mathbf{x}', \mathbf{r}) \leq \mathrm{t}$, eftersom det skulle betyda $\mathrm{d}(\mathbf{x}, \mathbf{x}') \leq \mathrm{d}(\mathbf{x}, \mathbf{r}) + \mathrm{d}(\mathbf{x}', \mathbf{r}) \leq 2\mathrm{t}$ (enligt triangelolikheten). Detta går ej eftersom $\mathrm{d}(\mathrm{C}) \geq 2\mathrm{t} + 1$, så **x** är det närmsta kodordet till **r**, och **r** avkodas på ett korrekt sätt, genom att man väljer **x** som det som skickats.

■

När linjära koder kommer på tal så används ofta notationen [n, k, d]. Bokstaven *n* svarar mot längden på koden, vidare står *k* för dimensionen av koden och slutligen *d* står för minsta avståndet som finns mellan kodorden. Det finns också en annan notation: (n, M, d) där n och

d är samma som ovan men där *M* i stället står för totala antalet kodord i den linjära koden. I detta fall har de sambandet att den linjära koden [n, k, d] är en kod av karaktären $(n, 2^k, d)$. Av detta så ser vi också att om dimensionen för en binär kod är k så är antalet kodord som kan matas fram 2^k. Har vi en annan typ av kod än den binära, där 2 byts ut mot ett annat tal, q, så får vi att den q-nära koden har q^k antal kodord. Detta ska tolkas som att vi har q^k element i V^n, dvs vektorrummet av k-tiplar över $\mathbb{Z}_q$, där q är ett primtal. Termerna här ska tolkas som att k står för längden av meddelandet och n står för längden av kodorden, där $n > k$. Hädanefter kommer jag mestadels använda mig av notationerna [n, k, d] och [n, k].

Om man vill öka det minsta avståndet för en känd linjär kod så kan man lägga till en övergripande paritetskontroll-siffra. Antag att C är en linjär (n, k, d)-kod. Vi kan då skapa en kod C', den utvidgande koden av C, genom att lägga till en paritetskontroll-bit till varje kodord $\mathbf{x} \in C$ för att då få kodorden $\mathbf{x}' \in C'$. För varje $\mathbf{x} = x_0x_1 \cdots x_{n-1} \in C$ låt varje $\mathbf{x}' = x_0x_1 \cdots x_{n-1}0$ om hammingtyngden av **x** är jämn, och låt $\mathbf{x}' = x_0x_1 \cdots x_{n-1}1$ om hammingtyngden av **x** är udda. Detta säkerställer att varje kodord i den utvidgade koden har en jämn hammingtyngd.

Sats 2: Låt C vara en linjär kod med minsta avståndet $d = d(C)$. Om d är udda, då är minsta avståndet i den utvidgade koden C' lika med $d + 1$ och om d är jämnt så är det minsta avståndet av C' lika med d.

Bevis:
För att visa detta behövs följande lemma.

Lemma 1: För en linjär kod så är $w(x + y) \equiv w(x) + w(y) \bmod 2$.

Bevis:

$$\mathbf{x} = x_1 \ldots x_n$$
$$\mathbf{y} = y_1 \ldots y_n$$

För varje plats i finns fyra möjligheter:

I) A: $\{i: x_i = 0 = y_i\}$
II) B: $\{i: x_i = 0, y_i = 1\}$
III) C: $\{i: x_i = 1, y_i = 0\}$
IV) D: $\{i: x_i = 1 = y_i\}$

där A, B, C och D betecknar alltså mängden av platser, där de olika situationerna inträffar. Nu är $w(x) = |C| + |D|$ och $w(y) = |B| + |D|$.

Vidare är $x_i + y_i = \begin{cases} 0 \text{ i fall I och IV, dvs om } i \in A \cup D \\ 1 \text{ i fall II och III, dvs om } i \in B \cup C. \end{cases}$

Alltså är $w(x + y) = |B| + |C|$.

Nu ser vi att

$w(x) + w(y) = |C| + |B| + 2|D| \equiv |C| + |B| \bmod 2$.

■

Nu vidare till beviset av sats 2.

Först ska vi visa att C′ är en linjär kod, d.v.s. att summan av 2 kodord

$x' = x_1 \dots x_{n-1}x_n$, och $y' = y_1 \dots y_{n-1}y_n$ också tillhör C'. Av definitionen följer att

$$x_n \equiv w(x) \bmod 2$$
$$y_n \equiv w(y) \bmod 2.$$

Om nu $x + y = z \in C$ så gäller enligt lemmat att $x_n + y_n \equiv w(x) + w(y) \equiv w(x + y) \equiv z_n$ (mod 2), och speciellt $x_n + y_n = z_n$ (där $z' = z_1 \dots z_{n-1}z_n$). Alltså adderas alla siffror mod 2 och koden är linjär. Sedan ska vi hitta minsta vikten i C'.

Vi har 2 fall: det första är att det finns ett element av minimal vikt i C' av typen $c' = c1$ (där $c \in C$ och sista siffran är 1 i elementet). Då gäller $w(c_1 1) \leq w(d1)$ för $d1 \in C'$, så $w(c1) \leq w(c) + 1 \leq w(d1) = w(d) + 1 \Longrightarrow w(c) \leq w(d)$ för alla d med udda vikt. För alla med jämn vikt har vi att $w(c_1 1) \leq w(d0)$, så $w(c) < w(c) + 1 = w(c1) \leq w(d0) = w(d)$. Detta medför att $w(c) < w(d)$. Sammantaget är alltså $c \in C$ ett element av minimal vikt i C och eftersom (c1) är ett element av minimal vikt i c' så är $d(c') = w(c1) = w(c) + 1 = d(c) + 1$. Eftersom $c1 \in C'$ så är $w(c)$ udda.

Det andra fallet är att alla element av minimal vikt i C' är av formen c0, alltså att sista siffran är 0. Då är

$$w(c0) \leq w(d0) \text{ om } d0 \in C',$$
$$\text{och } w(c0) < w(d1) \text{ om } d1 \in C'.$$[5]

I första fallet är $w(c) \leq w(d)$, och i andra fallet $w(c) < w(d) + 1 \Longrightarrow w(c) \leq w(d)$. Så även här är c ett element i C av minimal vikt och

$$d(C') = w(c0) = w(c) = d(c).$$

Notera att $d(c) = w(c)$ är udda eftersom $c0 \in C'$, enligt definitionen.

■

Kroppar, vektorrum och mer om linjära koder

För att ytterligare utveckla teorin om linjära koder behöver vi gräva djupare i såväl abstrakt som linjär algebra. En viktig struktur vi kommer använda oss av är teorin om *kroppar*. Exempel på detta kan vara det binära alfabetet eller modulo 2-räkning med addition och multiplikation, men låt oss se på den formella definitionen.

Sats 3:
Låt F vara en icke-tom mängd som är sluten under två binära operationer, + och *, dvs de operationerna matar in element a och b i F och matar ut andra element $c = a + b$ och $d = a * b$ i F. Då är $(F, +, *)$ en kropp om följande är uppfyllt:

1. Det finns en additiv enhet, ett element 0 i F så att $a + 0 = 0 + a = a$ för alla element $a \in F$.
2. Additionen är kommutativ: alltså att $a + b = b + a$, $\forall a, b \in F$.
3. Additionen är associativ: alltså att $(a + b) + c = a + (b + a)$, $\forall a, b, c \in F$.
4. Det finns en additiv invers för varje element: $\forall a \in F \; \exists$ ett element $-a$ i F så att $a + (-a) = 0$.

[5] Den strikta olikheten i andra fallet kommer av antagande att alla minimal vikt i C' är av formen c0.

5. Det finns en multiplikativ identitet betecknad 1 i F så att $a * 1 = 1 * a$ för alla element $a \in F$
6. Multiplikationen är kommutativ: $a * b = b * a \; \forall a, b \in F$
7. Multiplikationen är associativ: $a * (b * c) = (a * b) * c \; \forall a, b, c \in F$
8. Det finns en multiplikativ invers för alla nollskiljda element: för varje $a \neq 0$ i F extisterar det ett element betecknad a^{-1} i F så att $a * a^{-1} = a^{-1} * a = 1$.
9. Räkneoperationerna ovan är distributiva: $a * (b + c) = a * b + a * c$ och $(b + c) * a = b * a + c * a, \; \forall a, b, c \in F$

Eftersom vi antar att $0 \neq 1$ har vi att den minsta möjliga kroppen har två element, mer specifikt är detta det binära alfabetet. Denna kropp brukar betecknas $\mathbb{Z}_2$. Det finns också en alternativ beteckning $GF(2)$, vilket kommer från Galoiskroppen med två element, eftersom det på engelska kallas det för Galois field. Hädanefter använder jag mig dock av $\mathbb{Z}_2$.

Följande tabeller visar hur addition och multiplikation går till inom det binära alfabetet:

Tabell 2 – addition inom $\mathbb{Z}_2$:

+	0	1
0	0	1
1	1	0

Tabell 3 – multiplikation inom $\mathbb{Z}_2$:

×	0	1
0	0	0
1	0	1

Primtal spelar här en avgörande roll. Om vi har en uppsättning av heltal $\mathbb{Z}_p = \{0,1, \dots, p-1\}$, där p är ett primtal, kommer denna under addition av multiplikation mod p vara en kropp. Ifall p ej är ett primtal kommer det saknas en multiplikativ invers för alla element i mängden mod p. Befinner man sig i t.ex. $\mathbb{Z}_4 = \{0,1,2,3\}$ saknas det en multiplikativ invers till elementet 2, eftersom

$$2 \cdot 0 = 0 \neq 1$$
$$2 \cdot 1 = 2 \neq 1$$
$$2 \cdot 2 = 0 \neq 1$$
$$2 \cdot 3 = 2 \neq 1$$

så inget av $0, 1, 2, 3$ är en multiplikativ invers. Ett konkret exempel på när $\mathbb{Z}_p$ används är för ISBN-koder (streckkoderna som vanligtvis används vid tryckta medier), där det är just kropp-egenskaperna av $\mathbb{Z}_{11}$ som ger ISBN-koder sina feldetekterande egenskaper.[6]

[6] Adams, s. 19.

Definition 5: vektorrum
Låt F vara en kropp. En mängd V av element kallade vektorer utgör ett vektorrum om följande är uppfyllt för varje **u, v, w** ∈ V och för varje c, d ∈ F:

1. $\mathbf{v} + \mathbf{w} \in V$.
2. $c\mathbf{v} \in V$.
3. $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$.
4. $\mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{u} + \mathbf{v}) + \mathbf{w}$.
5. Det finns ett nollelement (också kallad nollvektor) $\mathbf{0} \in V$, så att $\mathbf{u} + \mathbf{0} = \mathbf{u}$.
6. För varje element $\mathbf{u} \in V$, finns det ett element betecknat $-\mathbf{u}$, så att $\mathbf{u} + (-\mathbf{u}) = \mathbf{0}$.
7. $c(\mathbf{u} + \mathbf{v}) = c\mathbf{u} + c\mathbf{v}$.
8. $(c + d)\mathbf{u} = c\mathbf{u} + d\mathbf{u}$.
9. $c(d\mathbf{u}) = (cd)\mathbf{u}$.
10. $1\mathbf{u} = \mathbf{u}$.

Definition 6:
Låt V vara ett vektorrum. En delmängd U i V kallas *underrum* av V om U också är ett vektorrum som lyder under samma definition för ett vektorrum.

Teorem 2:
Låt V vara ett vektorrum och F en kropp. En delmängd U i V är underrum om det
1. För varje par av vektorer $\mathbf{u}, \mathbf{v} \in U$ så finns $\mathbf{u} + \mathbf{v}$ också i U.
2. För varje $c \in F$ och $\mathbf{u} \in U$ så finns $c\mathbf{u}$ också i U.

Definition 7:
Två vektorer är *linjärt oberoende* om de inte är skalära multiplar av varandra. Om de är det så kallas de i sådana fall för *linjärt beroende*.

Exempel 3:
Två vanliga exempel på vektorrum är $\mathbb{R}^2$ och $\mathbb{R}^3$. För $\mathbb{R}^3$ kan vi se att vektorerna $(3,0,0)$ och $(0,2,0)$ är linjärt oberoende, eftersom de inte är skalära multiplar. Däremot om vi tar vektorerna $(1,2,3)$ och $(2,4,6)$ så ser vi att den sistnämnda är exakt en fördubbling av den första. Oftast är vi intresserade av vektorer som är oberoende alla andra vektorer i en mängd och inte bara mellan två stycken.

Exempel 4:
Om vi tar exempel 2 där mängden är {0000, 0101, 1010, 1111} och betraktar kodorden som vektorer ser vi att ingen av de tre sista är en skalär multipel av en annan. Ändå betraktar vi dem inte som linjärt oberoende eftersom det finns ytterligare ett krav för att uppfylla det, nämligen att de inte ska kunna skrivas som en kombination av varandra.

Definition 8:
Vektorer i en mängd kallas *linjärt oberoende* om inga av vektorerna är linjärkombinationer av några andra vektorer i mängden. Alltså; vektorerna $\{\mathbf{v_0}, \mathbf{v_1}, \dots, \mathbf{v_{n-1}}\}$ i en mängd är linjärt oberoende om den enda linjärkombinationen $c_0\mathbf{v_0} + c_1\mathbf{v_1} + \cdots + c_{n-1}\mathbf{v_{n-1}}$ som ger 0 är då alla skalärer $c_0, c_1, \dots, c_{n-1}$ är lika med 0. I annat fall kallas de för linjärt beroende.

Exempel 5:
Om vi än en gång återvänder till vektorrummet $\mathbb{R}^3$ har vi standardexemplet att vektorerna $(1,0,0)$, $(0,1,0)$ och $(0,0,1)$ är linjärt oberoende från varandra eftersom det enda sättet som $c_0(1,0,0), +c_1(0,1,0) + c_2(0,0,1) = 0$ är då $c_0 = c_1 = c_2 = 0$. Man ser också att hur man än lägger ihop två vektorer så går det inte att få den tredje av dessa. Om vi emellertid har vektorerna $(1,0,0)$, $(0,1,0)$ och $(2,2,0)$ så är den tredje en linjärkombination av de två första, eftersom de fördubblats och lagts ihop för att få den tredje vektorn. Man kan också skriva det på följande sätt: de värden på c_0, c_1, c_2 som löser $c_0(1,0,0), +c_1(0,1,0) + c_2(2,2,0) = 0$ är just $c_0 = 1, c_1 = 1$ och $c_2 = -2$ och detta medför att vektorerna är linjärt beroende av varandra, eftersom c_0, c_1, c_2 ej är noll.

Exempel 6:
För att beskriva mängden i exempel 2 ovan kan man ta fram de fyra kodorden med hjälp av endast tre stycken, t.ex. 0000, 0101, 1010, eftersom 0101+1010=1111 och därmed blir sista kodordet linjärt beroende av två andra.

Definition 9:
En bas är en mängd linjärt oberoende vektorer som *spänner upp* ett vektorrum, alltså att varje vektor i rummet kan uttryckas som en linjärkombination av basvektorerna. Som exempel kan vi ta föregående exempel med vektorerna $(1,0,0)$, $(0,1,0)$ och $(0,0,1)$, ty de utgör nämligen standardbasen för $\mathbb{R}^3$. Ett annat exempel kan vara $(1,2,3)$, $(1,3,5)$ och $(1,1,1)$. Huvudsaken här är att vektorerna ska vara linjärt oberoende, och spänna upp hela vektorrummet.

Definition 10:
Dimensionen av ett vektorrum V är antalet vektorer för basen av V.

För felkontrollerande koder är delrum av vektorrum ett viktigt begrepp. Om vi tar exempel från geometrin har vi att plan och linjer genom origo är delrum i det 3-dimensionella rummet. Linjerna är ett 1-dimensionellt vektorrum i $\mathbb{R}^3$ och plan är ett 2-dimensionellt vektorrum i $\mathbb{R}^3$.

Definition 11:
Låt V vara ett vektorrum över kroppen F och låt W vara en delmängd av V. Om W är ett vektorrum över F, så är W ett delrum av V.

Teorem 3:
Om W är en delmängd av ett vektorrum V över kroppen F, då är W ett delrum av V omm $\alpha\mathbf{u} + \beta\mathbf{v} \in W$ för alla $\alpha + \beta \in F$ och alla $\mathbf{u}, \mathbf{v} \in W$**.**

Bevis:
Anta att W är ett delrum av V, som är ett vektorrum över kroppen F. Då är W självt ett vektorrum över F. Alltså, enligt definitionen för ett vektorrum, $\mathbf{u} + \mathbf{v} \in W$ för alla $\mathbf{u}, \mathbf{v} \in W$ och $s\mathbf{v} \in W$ för alla $s \in F$. Det här medför att $\alpha\mathbf{u} + \beta\mathbf{v} \in W$ för alla $\alpha + \beta \in F$ och alla $\mathbf{u}, \mathbf{v} \in W$.

Anta nu att $\alpha\mathbf{u} + \beta\mathbf{v} \in W$ för alla $\alpha, \beta \in F$ och alla $\mathbf{u}, \mathbf{v} \in W$. Sätter vi då $\alpha = \beta = 1 \in F$ ger oss det första kriteriet för ett vektorrum och sätter vi $\alpha = 1 \in F$ och $\beta = 0 \in F$ så får vi det andra kriteriet.

■

Varför är detta intressant för kodteori? Jo, för linjära koder är i själva verket delrum av vektorrum och här är kanske den tydligaste kopplingen mellan kodteori och linjär algebra.

Definition 12:
En linjär kod av längd n över $\mathbb{Z}_q$ är ett delrum av vektorrummet $\mathbb{Z}_q^n = V^n$.

Av detta följer också att om C är en linjär kod så kommer linjärkombinationen av vilka kodord som helst i C också vara i C. Eftersom C är ett delrum av V^n, så stämmer det enligt föregående teorem, eftersom den uppfyller kriterierna nämligen att $\mathbf{u} + \mathbf{v} \in C$ för alla $\mathbf{u}, \mathbf{v} \in C$ och att $\alpha\mathbf{u} \in C$ och $\alpha \in \mathbb{Z}_q$.

Definition 13:
Dimensionen av en linjär kod C av längd n är dess dimension som ett delrum av $\mathbb{Z}_q^n = V^n$.

Exempel 7:
Återigen kan vi ta exemplet med koden C= {0000, 0101, 1010, 1111}. I detta fall är kroppen $\mathbb{Z}_2$. Som vi tidigare såg kan 1111 bildas med hjälp av kodorden 0101 och 1010, vilket gör att denna kod spänns upp av dessa två kodord. Eftersom de är två till antalet blir dess dimension precis $k = 2$. Vi kan också se att antalet kodord precis är $2^2 = 4$. Denna kod kan alltså skrivas på formen $[4, 2, 2]$.

Att skapa linjära koder: generatormatriser

För att göra om meddelanden till kodord krävs något som kallas för *generatormatris*.

Exempel 8 - generatormatris:
Säg att vi har en [3,2] linjär kod C, alltså ett 2-dimensionellt delrum av vektorrummet V^3 över $\mathbb{Z}_2$. Då kan C spännas upp av t.ex. de två linjärt oberoende vektorerna 011 och 110. Detta ger oss generatormatrisen

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

Genom att ta alla linjärkombinationer av raderna i G, så kan följande kod genereras:

$$C = \{110,011,101,000\}$$

Vi behöver dock inte själva försöka lista ut alla linjärkombinationer, utan dessa kodord kan man också metodiskt få fram genom att ta alla binära 2-tiplar och multiplicera med generatormatrisen:

$$[00]\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} = [000]$$
$$[01]\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} = [011]$$
$$[10]\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} = [110]$$
$$[11]\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} = [101]$$

Med denna matris kan man alltså få fram alla möjliga kodord som finns i koden. Detta bidrar till enkelhet och förutsägbarhet, vilket är styrkorna med dessa linjära koder. Mer specifikt har

vi att en $k \times n$ generatormatris G för en $[n, k]$ linjär kod C ger ett kompakt sätt att beskriva alla kodord i koden. För att koda ett meddelande, **m**, av längden k tar man och multiplicerar **m** med G för att få ett kodord av längden n. Detta är en funktion $\mathbf{m} \to \mathbf{m}G$ som avbildar vektorrummet V^n till ett k-dimensionellt delrum, som i detta fall är koden självt C. Allt detta kan kokas ner till följande definition.

Definition 14:
En $k \times n$ matris G, vars k st rader av längd n utgör en bas för en $[n, k]$ linjär kod C kallas för en *generatormatris* av koden C.

Paritetscheckmatriser

Definition 15:
Låt C vara en $[n, k]$ linjär kod. En *paritetscheckmatris* för C är en $(n - k) \times n$ matris H så att $\mathbf{c} \in C$ om och endast om $\mathbf{c}H^T = 0$.

Sats 4:

En paritetscheckmatris till $C = [I_k|P]$ är $H = [-P^T, I_{n-k}]$.

Bevis: $H^T = \begin{bmatrix} (-P^T)^T \\ I_{n-k}^T \end{bmatrix} = \begin{bmatrix} -P \\ I_{n-k} \end{bmatrix}$

och

$CH^T = I_k \cdot (-P) + P \cdot I_{n-k} = -P + P = 0.$

Det visar att om c tillhör den linjära koden som C definierar, och därmed är en rad i C) så är $cH^T = 0$.

Därefter behöver vi visa att $cH^T = 0 \Rightarrow$ d tillhör den linjära koden som c definierar.

Eftersom $d = (d_1 \dots d_k, d_{k+1} \dots d_{n-k})$ har vi $dH^T = (d_1 \dots d_k, d_{k+1} \dots d_{n-k}) \begin{pmatrix} -P \\ I_{n-k} \end{pmatrix}$

$$= (d_1 \dots d_k) \cdot (-P) + (d_{k+1} \dots d_{n-k}) I_{n-k}$$
$$= -(d_1 \dots d_k) \cdot (P) + (d_{k+1} \dots d_{n-k})$$

Så $dH^T = 0 \Leftrightarrow (d_{k+1}, \dots d_n) = (d_1 \dots d_k)P$. (*)

Vidare har vi att $(d_1 \dots d_k)I_k = (d_1 \dots d_k)$ och detta tillsammans med (*) ger att:

$$(d_1 \dots d_k, d_{k+1} \dots d_n) =$$
$$= (d_1 \dots d_k)(I_k|P)$$
$$= (d_1 \dots d_k)c$$

Detta innebär att d är en linjärkombination av raderna i C och alltså tillhör den linjära koden.

■

Exempel 9:
Säg att vi har koden $\mathrm{C} = \{111,000\}$. En paritetscheckmatris till den är då

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

För att se att denna fungerar till koden kan vi beräkna produkten av $\mathbf{c}\mathrm{H}^{\mathrm{T}}$ för ett godtyckligt kodord $\mathbf{c} = [\mathrm{x}, \mathrm{y}, \mathrm{z}]$ och se att produkten verkligen blir noll.

$$[\mathrm{xyz}] \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} = [\mathrm{x} + \mathrm{z}, \mathrm{y} + \mathrm{z}]$$

Om $\mathrm{x} + \mathrm{z} = 0$ och $\mathrm{y} + \mathrm{z} = 0$ har vi (mod 2) att $\mathrm{x} = -\mathrm{z} = \mathrm{z}$ och $\mathrm{y} = -\mathrm{z} = \mathrm{z}$, vilket medför att $\mathrm{x} = \mathrm{y} = \mathrm{z}$. Detta i sin tur ger kodorden 000 respektive 111.

Exempel 10:
I exempel 9 hade vi ett exempel på en repetitionskod, (en kod med kodord som bara innehåller antingen ettor eller nollor) av längd 3, och en paritetscheckmatris till repetitionskoden av längd 5, dvs $\mathrm{C} = \{11111,00000\}$, kan alltså vara:

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

Om vi återigen tar ett godtyckligt kodord: $\mathbf{c} = [\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e}]$ kan vi erinra oss om att produkten blir noll:

$$[\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e}] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [\mathrm{a} + \mathrm{b}, \mathrm{b} + \mathrm{c}, \mathrm{c} + \mathrm{d}, \mathrm{d} + \mathrm{e}]$$

Om $\mathrm{a} + \mathrm{b} = 0,\ \mathrm{b} + \mathrm{c} = 0,\ \mathrm{c} + \mathrm{d} = 0,\ \mathrm{d} + \mathrm{e} = 0$ är (mod 2) $\mathrm{a} = -\mathrm{b} = \mathrm{b},\ \mathrm{b} = -\mathrm{c} = \mathrm{c},\ \mathrm{c} = -\mathrm{d} = \mathrm{d},\ \mathrm{d} = -\mathrm{e} = \mathrm{e}$ så $\mathrm{a} = \mathrm{b} = \mathrm{c} = \mathrm{d} = \mathrm{e}$ vilket ger möjligheterna 00000 och 11111.

För just repetitionskoden av längd 5 fungerar denna paritetscheckmatris, eftersom båda kodorden är precis de kodord som uppfyller $\mathbf{c}\mathrm{H}^{\mathrm{T}} = 0$.

Vidare har vi att en paritetscheckmatris uppfyller $\mathrm{GH}^{\mathrm{T}} = 0$, eftersom GH^{T} består av G:s rader multiplicerade av H^{T}, och eftersom dessa rader är element i koden har vi att:

$$\mathrm{GH}^{\mathrm{T}} = 0.$$

Det finns en specifik relation mellan en generatormatris och en paritetscheckmatris, nämligen att den ena går att härleda från den andra och omvänt. Om vi har en generatormatris för en kod på formen $[\mathrm{n}, \mathrm{k}]$ som är i standardform, likt

$$\mathrm{G} = [\mathrm{I}_{\mathrm{k}} | \mathrm{P}]$$

då finns alltid en paritetscheckmatris H genom

$$\mathrm{H} = [-\mathrm{P}^{\mathrm{T}} | \mathrm{I}_{\mathrm{n-k}}]$$

som vi visade i sats 4.

Exempel 11:
Om en kod har generatormatrisen

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

kommer dess paritetscheckmatris att vara

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

Sammanfattningsvis för detta avsnitt kan vi säga att paritetscheckmatriser är ett sätt att kontrollera om ett kodord faktiskt tillhör koden eller ej, och den används också vid avkodning. Mer om detta i nästa avsnitt.

Definition 16: *nollrummet* av en matris H är mängden av alla vektorer **x** som gör att $H\mathbf{x}^T = 0$.

Inom linjär algebra talar man ofta om nollrummet då man ska hitta kärnan (eng. kernel) för linjära avbildningar. Här finns dock en koppling till linjära koder.

Sats 5: Alla kodord i C utgör nollrummet för dess paritetscheckmatris H.

Bevis: För paritetscheckmatrisen H för en kod C har vi att $\mathbf{c}H^T = 0 \iff \mathbf{c} \in C$, därmed är $0^T = (\mathbf{c}H^T)^T = H\mathbf{c}^T$. Alltså har vi att alla kodord i C utgör nollrummet för dess paritetscheckmatris H.

■

Paritetscheckmatriser och linjär avkodning

Detta avsnitt handlar om hur paritetscheckmatriser, tillsammans med sidoklasser (eng. cosets), används vid linjär avkodning, och denna avkodning använder sig av en ”närmaste grannen”-metod. Denna metod bygger på att om ett kodord blir fel byter man till det förmodade kodordet med minst hammingavstånd till det erhållna kodordet. Den avgörande ekvationen fås från en paritetscheckmatris (dvs $\mathbf{c}H^T = 0 \iff \mathbf{c} \in C$) och används vid det första avkodningssteget. Då ett kodord **r** erhålles så beräknas produkten $\mathbf{r}H^T$. Om produkten är 0 är det inga fel med kodordet **r** och vi behåller det som det är och gör ingen ytterligare korrigering.

Definition 17:
Anta att C är en $[n, k]$ linjär kod över $\mathbb{Z}_q$ och anta att **a** är en godtycklig vektor i V^n. Då är mängden $\mathbf{a} + C = \{\{\mathbf{a} + \mathbf{x}\} \mid \mathbf{x} \in C\}$ en *sidoklass* till C.

Teorem 4:
Anta att C är en $[n, k]$ linjär kod över $\mathbb{Z}_q$. Då finns varje vektor av V^n i någon sidoklass av C, varje sidoklass innehåller precis q^k vektorer, och två sidoklasser är antingen disjunkta eller lika.

Bevis:

1) Disjunkthet

Antag att $a + C \cap a' + C \neq \emptyset$. Då finns $a + c_1 = a' + c_2$ som ligger i bägge sidoklasserna. Men V^n är ett vektorrum så det ger att $a = a' + (c_2 - c_1) \Rightarrow a \in a' + C$. Alltså är också $a + c = a' + c + (c_2 - c_1) \in a' + C$ för alla $c \in C$. Således är hela $a + c \subset a' + c$ och på samma sätt så $a' + C \subset a + C$, d.v.s. de är lika.

2) Alla sidoklasser har lika många element

Antag att vi har $a + C$ och $a' + C$. Definiera en avbildning

$$a + C \to a' + C$$

genom

$$a + c \to a' + c$$

för $c \in C$.

Påstående: avbildningen är bijektiv.

Antag att $a' + c_1 = a' + c_2$, vilket medför att $a' \neq c_1 - a' = a' + c_2 - a' \Leftrightarrow c_1 = c_2 \Longrightarrow a + c_1 = a + c_2$. Så avbildningen är injektiv och även surjektiv, eftersom allting i $a' + C$ ser ut som $a' + c$ och alltså är bild av $a + c$. Därmed så har de två sidoklasserna samma antal element som i $0 + C = C$, dvs i q^k.

Vi räknar nu ut antalet element i
$V^n = q^n = \text{antalet sidoklasser} \cdot \text{antalet element i en sidoklass}$ och
$q^n = \text{antalet sidoklasser} \cdot q^k$, vilket medför att antalet sidoklasser är q^{n-k}.

■

Exempel 12:
Låt *C* vara en $[4,2]$ linjär kod med generatormatrisen

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

Av denna får vi koden med följande kodord: $C = \{0000, 1011, 0101, 1110\}$ och sidoklasserna till C ges av:

$$0000 + C = \{0000, 1011, 0101, 1110\} = 1011 + C = 0101 + C = 1110 + C$$
$$1000 + C = \{1000, 0011, 1101, 0110\} = 0011 + C = 1101 + C = 0110 + C$$
$$0100 + C = \{0100, 1111, 0001, 1010\} = 1111 + C = 0001 + C = 1010 + C$$
$$0010 + C = \{0010, 1001, 0111, 1100\} = 1001 + C = 0111 + C = 1100 + C$$

Här ser vi ett exempel på att sidoklasser kan vara lika, eftersom $0001 + C$, ger precis samma kodord som $0100 + C$. Sidoklasserna, $\mathbf{a} + C$, kan alltså beskrivas med olika representanter $\mathbf{a}'$. Givetvis vill vi ha en sådan enkel beskrivning som möjligt för dem.

Definition 18:
En *sidoklassledare* av en sidoklass väljs ut att vara en av vektorerna med minst tyngd i sidoklassen.

Sidoklassalgoritmen av en $[n, k]$ linjär kod fungerar på så sätt att man delar in $\mathbb{Z}_q^n$ i sidoklasser av en kod C. Det finns $\frac{q^n}{q^k} = q^{n-k}$ sidoklasser, där varje sidoklass innehåller q^k element. För varje sidoklass ska man välja ut en sidoklassledare, där man ibland får välja en

ledare godtyckligt (se exempel ovan med $0001 + \mathrm{C}$ och $0100 + \mathrm{C}$). Om vi erhåller meddelandet **r**, letar vi efter sidoklassledaren som innehåller **r**. Denna sidoklass är på formen $\mathbf{e} + \mathrm{C}$, och vi förmodar att $\mathbf{r} - \mathbf{e}$ var det som sändes. Här spelar sidoklassledarna rollen som felvektor som pekar på var felet är, vilket är fördelaktigt eftersom ledarna är per definition den vektor med minst tyngd och de felvektorer med minst tyngd är de som är mest troliga.

Exempel 13:
Anta att vi erhåller kodordet 1001, om vi använder sidoklasserna i föregående exempel, då ser vi att det erhållna kodordet är på samma rad som sidoklassledaren 0010, därmed ska vi ta $1001 - 0010$ och kodordet som från början skickades var 1011.

För att tillämpa denna typ av avkodning gör man en standardarray, där man listar alla sidoklasser radvis. Längst till vänster placerar vi sidoklassledaren. På första raden har vi också alltid $\mathbf{0} +$ alla kodorden i C., med sidoklassledaren **0** längst till vänster. För nästkommande rader tar man de vektorer i $\mathbb{Z}_q^n$ som inte finns i C med lägst tyngd och därefter adderar den till den fösta raden för att få fram resterande vektorer på samma rad. Detta gör man för samtliga sidoklassledare. Standardarrayen för föregående exempel ser ut som följande:

0000	1011	0101	1110
1000	0011	1101	0110
0100	1111	0001	1010
0010	1001	0111	1100

Om man erhåller kodordet **r** kommer det korrigerade kodordet att finnas längst upp i samma kolonn. Mer om detta i exemplet nedan. De felvektorer som kommer korrigeras är precis sidoklassledarna och eftersom dessa har lägst tyngd medför det att standardarrayen faktiskt avkodar enligt ”närmsta-granne”-metoden.

Syndromavkodning är ett närbesläktat avkodningsschema som använder sig av en paritetscheckmatris H av en kod. Låt oss anta att **x** är meddelandet som sänds och att **r** är det som mottas. Ett *syndrom* till **r** beräknas genom $\mathbf{r}\mathrm{H}^{\mathrm{T}}$och betecknas med $\mathrm{S}(\mathbf{r})$.

Sats 4 säger i den här kontexten att för syndromet $\mathrm{S}(\mathbf{r})$ och en paritetscheckmatris H har vi att $\mathrm{S}(\mathbf{r}) = \mathbf{r}\mathrm{H}^{\mathrm{T}} = 0$ omm $\mathbf{r} \in \mathrm{C}$.

Om $\mathrm{S}(\mathbf{r}) = \mathbf{r}\mathrm{H}^{\mathrm{T}} = 0$ antar vi att det troligen inte skett något fel. Om tvärtom $\mathbf{r}\mathrm{H}^{\mathrm{T}} \neq 0$ vet vi att åtminstone ett fel inträffat och $\mathbf{r} = \mathbf{x} + \mathbf{e}$, där **x** är ett kodord och **e** en felvektor. Eftersom $\mathbf{x}\mathrm{H}^{\mathrm{T}} = 0$ har vi att $\mathbf{r}\mathrm{H}^{\mathrm{T}} = (\mathbf{x} + \mathbf{e})\mathrm{H}^{\mathrm{T}} = \mathbf{x}\mathrm{H}^{\mathrm{T}} + \mathbf{e}\mathrm{H}^{\mathrm{T}} = 0 + \mathbf{e}\mathrm{H}^{\mathrm{T}} = \mathbf{e}\mathrm{H}^{\mathrm{T}}$.

Exempel 14:
Om vi tar föregående standardarray och utvidgar för att använda syndromavkodning ser den ut på följande vis:

0000	1011	0101	1110	00=S(0000)
1000	0011	1101	0110	11=S(1000)
0100	1111	0001	1010	01=S(0100)
0010	1001	0111	1100	10=S(0010)

Syndromavkodningsalgoritmen fungerar på följande sätt: när en vektor **r** erhålles så beräknas syndromet $\mathrm{S}(\mathbf{r}) = \mathbf{r}\mathrm{H}^{\mathrm{T}}$. Leta efter syndromet i tabellen. Avkoda **r** som $\mathbf{r} - \mathbf{e}$ där **e** är

sidoklassledaren i den rad som innehåller $S(\mathbf{r})$. Detta är ekvivalent till att avkoda **r** som det kodord på toppen i den kolonn som innehåller **r**.

Med syndrom sparar man både tid och utrymme, eftersom man bara behöver lagra felvektorerna och dess syndrom, och inte hela standardarrayen. När n blir stort så tar det mycket lång tid och kan vara svårt att finna vektorn i standardarrayen, till skillnad från att finna syndromet i en syndromkolonn.

Exempel 15:
Om vi använder ovanstående standardarray och antar att vi fått 1111 som kodord, och när vi beräknar syndromet fås $S(1111) = 01$. Detta syndrom återfinns på den tredje raden i syndromkolonnen och här har vi att sidoklassledaren är 0100, vilket också är elementet med minsta vikt i den sidoklassen, varför vi avkodar 1111 som $1111 - 0100 = 1011$. Detta går också att se i standardarrayen, eftersom 1011 är längst upp i kolonnen, ovanför 1111.

Teorem 5: för en kod gäller att syndromet av ett erhållet kodord är en vektor som är lika med summan av de kolonner i H som motsvarar positionerna där felen inträffade.

Bevis: Anta att **r** är ett erhållet kodord med syndromet $S(\mathbf{r}) = \mathbf{r}H^T$. Eftersom $\mathbf{r} = \mathbf{c} + \mathbf{e}$, där **c** är ett korrekt kodord och **e** är en felvektor, har vi att:

$$S(\mathbf{r}) = \mathbf{r}H^T = (\mathbf{c} + \mathbf{e})H^T = \mathbf{c}H^T + \mathbf{e}H^T = \mathbf{0} + \mathbf{e}H^T = \mathbf{e}H^T.$$

Det nästsista steget följer från det att H är en paritetscheckmatris till C. Eftersom $\mathbf{e}H^T$ är en kombination av kolonnerna i H som motsvarar de positionerna där felen uppstod så stämmer sambandet.

■

Paritetscheckmatriser, minsta avstånd och singleton-gräns

Förutom ovanstående användning av paritetscheckmatriser så kan de också användas för att hitta det minsta avståndet i en kod.

Teorem 6:
Låt C ha en paritetscheckmatris H. Det minsta avståndet av C är lika med minsta nollskiljda antalet kolonner i H för vilken en icketrivial linjärkombination av kolonnerna summeras ihop till noll.

Bevis:
Eftersom H är en paritetscheckmatris för C, så gäller att $\mathbf{c} \in C$ omm $0 = \mathbf{c}H^T$. Låt kolonnvektorerna till H vara $\{d_0, d_1, \dots, d_{n-1}\}$. Matrisekvationen $0 = \mathbf{c}H^T$ kan uttryckas på följande sätt:

$$0 = \mathbf{c}H^T = (c_0, c_1, \dots, c_{n-1})[\mathbf{d_0}\mathbf{d_1} \cdots \mathbf{d_{n-1}}]^T = c_0\mathbf{d_0} + c_1\mathbf{d_1} + \cdots + c_{n-1}\mathbf{d_{n-1}}$$

Detta visar att **c** är ett kodord med tyngden $d > 0$ omm det finns en icketrivial linjärkombination av d kolonner i H som blir noll. Det följer från sats 1 att det minsta avståndet av C är lika med det minsta nollskilda antalet kolonner i H för vilken en icketrivial linjärkombination av kolonnerna blir noll.

■

Exempel 16:

Anta att vi har en paritetscheckmatris $H = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ för en kod C.

Med hjälp av teorem 6 kan vi då ta reda på det minsta avståndet för koden genom att studera kolonnerna och deras linjärkombinationer. Eftersom ingen kolonn är 0 kan det minsta avståndet inte vara 1. Ej heller är summan av några två kolonner lika med 0, så minsta avståndet är inte 2. Den tredje kolonnen är dock en linjärkombination av kolonn 1 och 2, så det minsta avståndet för koden är 3 eftersom vi behöver summera dessa kolonner för att det ska bli noll.

Teorem 7: Singleton-gräns
Det minsta avståndet d för en $[n, k]$ linjär kod uppfyller $d \leq n - k + 1$.

Bevis:
En $[n, k]$ linjär kod har en paritetscheckmatris H bestående av $(n - k)$ linjärt oberoende rader, och $(n - k)$ linjärt oberoende kolonner. Vilken samling som helst av $(n - k + 1)$ i H måste vara linjärt beroende. Av föregående teorem följer nu att det minsta avståndet uppfyller villkoret i teoremet.[7]

■

Koder som uppfyller Singleton-gränsen med likhet kallas för maximala avståndsseparabeln, eng. ”maximum distance separable”. Utifrån parametrarna n och k ger de det bästa minsta avståndet.

Hammingkoder

Att syndromet av en erhållen vektor är lika med summan av kolonnerna av paritetscheckmatrisen H där fel uppstod (teorem 5) ger en ledtråd om hur man kan skapa en paritetscheckmatris för en kod där sidoklass- eller syndromavkodning kommer genomföras. Kolonnerna för H ska vara nollskilda, eftersom ett fel på motsvarande position inte skulle upptäckas av syndromavkodaren. Vidare har vi att kolonnerna i H måste vara distinkta eftersom om två kolonner är samma så går det inte att skilja på fel i dessa två positioner.

Med detta som utgångspunkt skapar vi något som kallas för *hammingkoder*, $\mathbf{H_r}, r \geq 2$ och varje paritetscheckmatris har precis r rader, vilket leder till att varje kolonn i matrisen har längden r. Det finns precis $2^r - 1$ nollskilda vektorer av längden r och för att skapa en paritetscheckmatris för H_r används precis alla dessa $2^r - 1$ vektorer.

Definition 19 - hammingkoder:
En *hammingkod* $\mathbf{H_r}$ av längden $n = 2^r - 1, r \geq 2$, har paritetscheckmatrisen H vars kolonner består av alla nollskilda vektorer av längd r och de används precis en gång. Detta ger oss en linjär med längden $n = 2^r - 1$ och dimensionen $k = 2^r - 1 - r$.

[7] S. 32 Adams

En paritetscheckmatris för en [7,4,3] hammingkod $\mathbf{H_3}$ kan se ut på följande sätt, och här ordnas kolonnerna i stigande ordning:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Vi ändrar paritetscheckmatrisen H till standardform genom att omgruppera kolonnerna:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

Vi identifierar att de tre första kolonner bildar en enhetsmatris för sig, och för de resterande fyra kolonnerna ska vi hitta transponatet för att skapa en generatormatris:

$$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}.$$

Om vi till denna matris lägger till enhetsmatrisen på höger sida får vi en generatormatris G på standardform:

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Exempel 17 – kodning med hammingkoder:
Säg att vi vill koda meddelandena 0000, 1100 och 0101, då får vi:

$$(0000)\mathrm{G} = (0000000)$$
$$(1100)\mathrm{G} = (1101100)$$
$$(0101)\mathrm{G} = (0100101)$$

Vi testar att dessa kodord stämmer genom att multiplicera med H^{T}, där

$$\mathrm{H}^{\mathrm{T}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$(0000000)\mathrm{H}^{\mathrm{T}} = (000)$$
$$(1101100)\mathrm{H}^{\mathrm{T}} = (000)$$
$$(0100101)\mathrm{H}^{\mathrm{T}} = (000)$$

När vi avkodar hammingkoder erhåller vi en vektor $\mathbf{r}$, beräkna då syndromet $\mathrm{S}(\mathbf{r})$. Om vi får $\mathrm{S}(\mathbf{r}) = 0$ antar vi att $\mathbf{r}$ faktiskt var det kodord som skickades. Om $\mathrm{S}(\mathbf{r}) \neq 0$, och vi antar att det bara är ett fel, har vi att $\mathrm{S}(\mathbf{r})$ är lika med den kolonn i H som motsvarar koordinaten där $\mathbf{r}$ inträffade. Hitta då den kolonn som hör till $\mathrm{S}(\mathbf{r})$ och rätta felet. Om vi exempelvis säger att vi fått $\mathbf{r} = (1000000)$, alltså skiljer sig på den första biten jämfört med kodordet (0000000), och multiplicerar den med transponatet får vi att $\mathrm{S}(1000000) \neq 0$:

$$(1000000)\mathrm{H}^{\mathrm{T}} = (100)$$

Den första kolonnen i H är precis $\begin{pmatrix}1\\0\\0\end{pmatrix}$ och detta visar alltså att felet ligger på den första biten,

och när vi ändrar den fås precis (0000000), vilket är ett av våra korrekta kodord.

Ibland kan man utvidga koden och lägga till en paritetscheckbit på en redan existerande kod för att ytterligare stärka koden. Här nedan har vi ett exempel på en paritetscheckmatris för en utvidgad hammingkod $\mathbf{H_3}$**:**

$$\begin{bmatrix}0&1&1&1&1&0&0&0\\1&0&1&1&0&1&0&0\\1&1&0&1&0&0&1&0\\1&1&1&1&1&1&1&1\end{bmatrix}.$$

Eftersom en kod kan rätta upp till t fel enligt $\mathrm{d}(\mathrm{C}) \geq 2\mathrm{t}+1$, där d står för det minsta hammingavståndet i koden, har vi att denna kod kan rätta ett fel.

Cykliska koder

Följande stycken är baserade på ”Introduction to Cryptography – with Coding Theory” av Trappe & Washington, s. 426 och framåt.

En kod C kallas cyklisk om ett kodord $(\mathrm{c}_1, \mathrm{c}_2, \ldots, \mathrm{c}_\mathrm{n})$ finns i C så betyder det att även $(\mathrm{c}_\mathrm{n}, \mathrm{c}_1, \mathrm{c}_2, \ldots, \mathrm{c}_{\mathrm{n}-1})$ också finns i C.

Exempelvis har vi att om $(1,1,0)$ finns i en viss cyklisk kod så tillhör även $(0,1,1)$ och $(1,0,1)$ den koden. Alla möjliga cykliska permutationer av ett kodord är alltså ett kodord. Denna cykliska egenskap har flera fördelar, där den främsta fördelen är att koden får en struktur som är enkel att följa.

Som ett annat exempel kan vi anta att vi har följande generatormatris:

$$\mathrm{G} = \begin{pmatrix}1011100\\0101110\\0010111\end{pmatrix}$$

där raderna skapar ett tredimensionellt delrum till ett sjudimensionellt rum. Faktum är att alla kodorden i denna [7,3,4] kod, förutom kodordet med idel nollor, kan fås genom permutationer av den första raden.

$$\mathrm{G} = \{(0,0,0,0,0,0,0), (1,0,1,1,1,0,0), (0,1,0,1,1,1,0), (0,0,1,0,1,1,1),$$
$$(1,0,0,1,0,1,1), (1,1,0,0,1,0,1), (1110010), (0111001)\}$$

Det går även att ta fram kodorden på algebraiskt vis och den metoden är central för så kallade BCH-koder, vilken är en typ av cyklisk kod. Metoden bygger på att de olika ettorna i kodorden kan representeras som koefficienter för ett polynom, oftast modulo 2. För detta exempel kan vi låta $\mathrm{Z}_2[\mathrm{X}]$ vara polynom i variabeln X med bas $1, \mathrm{x}, \mathrm{x}^2, \ldots, \mathrm{x}^{\mathrm{n}-1}$ och med

koefficienter modulo 2 och därefter blir $Z_2[X]/(X^7-1)$ dessa polynom modulo (X^7-1) med bas $1, x, x^2, \ldots, x^6$. När vi då får ett polynom med grad större än eller lika med 7 så dividerar vi det polynomet med (X^7-1) och matar ut resten.

Eftersom den första raden i matrisen ovan består av (1011100) kan vi skriva om det som polynomet $g(x) = 1 + X^2 + X^3 + X^4$. För att få fram alla kodorden ska vi ta fram alla produkter av $g(X)f(X) = a_0 + a_1X + \ldots + a_6X^6$, där koefficienterna av produkten beskrivs som en vektor $(a_0, \ldots, a_6)$.

Lägg märke till att $xg(x), x^2g(x), x^3g(x), \ldots, x^7g(x)$ alla motsvarar kodord, enligt definitionen. Exempelvis har vi att $xg(x) = x(1 + X^2 + X^3 + X^4) = x + x^3 + x^4 + x^5$ är den första permutationen ovan, nämligen $(1,0,1,1,1,0,0)$ och $x^2g(x) = x^2 + x^4 + x^5 + x^6$ vilket motsvarar $(0,0,1,0,1,1,1)$ och så vidare. För de som får högre grad 6 räknar vi ut det på följande sätt:

$$x^5(1 + x^2 + x^3 + x^4) = x^5 + x^7 + x^8 + x^9$$

vilket vi tar och dividerar med $X^7 - 1$:

	x^2	x	1	
x^9	x^8	x^7	x^5	$x^7 - 1$
x^9	x^2			
x^8	x^7	x^5	x^2	
x^8	x			
x^7	x^5	x^2	x	
x^7	1			
x^5	x^2	x	1	

och detta motsvarar (1110010). Sedan kan vi ta linjärkombinationer av kodorden $xg(x)$ och så få alla kodord som svarar mot $f(x)g(x)$.

Cykliska koder kan med fördel användas i så kallade skiftregister, vilka är användbara där data ska skickas med höghastigheten som till exempel i fiberoptik.[8]

Reed-Solomon-koder

En mer avancerad form av självrättande koder är en cyklisk kod som kallas för Reed-Solomon-koder, ofta förkortad som RS-koder. Namnet kommer från dess skapare Irving S. Reed och Gustave Solomon. Den stora skillnaden mellan Reed-Solomon-koder och hammingkoder är att den förstnämnda kodar grupper av bits, medan hammingkoder kodar en bit i taget.[9] För RS-koder brukar också *irreducibla polynom* och *primitiva polynom* komma på tals. Ett irreducibelt polynom som är det minsta polynomet av ett primitivt element i en kropp kallas för primitivt polynom.[10] Vi går inte in närmare på detta utan ska visa hur det fungerar med ett exempel nedan.

[8] Sarah Spence Adams, s. 38.
[9] Moreira, Farrell, s. 1.
[10] Van Lint, s. 10.

Reed-Solomon koder används vid lagring som CD-skivor men har också använts flitigt av NASA vid djuprymdskommunikation. Exempelvis hjälpte dessa koder till att få hem bilder från Saturnus och Neptunus vid *Voyager*-projektet. [11]

Reed Solomon-koder används också vid feldetektering och felkorrigering vid ADSL-/VDSL-linjer, DVD-spelare och i många andra typer av datalagring och digital kommunikation.[12] En anledning RS-koder är att den är effektiv på att behandla så kallade klusterfel, alltså flera fel som ligger nära varandra. Detta händer just ofta i praktiken vid användning av exempelvis telekommunikation och lagringssystem.[13] En invändning mot att använda det är att det kan upplevas som beräkningstungt. Dock finns det olika algoritmer att tillgå vilket möjliggör beräkningar med dator.

Definition 21:
En Reed-Solomon-kod är en cyklisk kod med längden $n = q - 1$ över $\mathbb{F}_q$ (se nedan för en definition). Kodorden svarar alltså mot polynom i $\mathbb{F}_q[x]/(x^n - 1)$, där $\mathbb{F}_q$ är en kropp med $q = p^\alpha$ element. Dess generator för den typen av kod är $g(x) = \prod_{i=1}^{d-1}(x - \alpha^i)$, där α är en primitiv i $\mathbb{F}_p$.[14] En kropp med q element har ett primitivt element α så att $\{1, \alpha, \dots, \alpha^{q-1} = 1\}$ är alla element i $\mathbb{F}_q$ utom 0.

RS-kodning

Följande stycken är hämtade från ”Reed Solomon Explained” av Tony Hill, s. 10 och framåt.[15]

För att enklare förstå RS-koder behöver läsaren vara förtrogen med ändliga kroppar. För varje primtalspotens p^n finns det en kropp $\mathbb{F}_q$ med precis q element. Till exempel har vi att $\mathbb{F}_8$ är $\mathbb{Z}_2[x]/(x^2 + x + 1) = \{\alpha_0 + a_1 x + a_2 x^2;\ a_0, a_1, a_2 \in \mathbb{Z}_2\}$. Vidare har vi att de 8 elementen i $\mathbb{F}_8$ svarar mot de 8 första heltalen $0, 1, \dots, 7$ i binär representation genom att polynomet $\alpha_0 + a_1 x + a_2 x^2$ svarar mot det binära talet $a_2 a_1 a_0$. I tabellerna nedan beskrivs addition och multiplikation med hjälp av notationen $0, 1, \dots, 7$. För såväl additionen som multiplikationen kommer den ske som vanligt, men vid multiplikation kommer resultatet delas med $x^2 + x + 1$.

Precis som för tidigare nämnda koder har vi för RS-koder att en bit av kodordet representerar meddelandet man vill skicka, medan en annan (mindre) del står för de självrättande egenskaperna. T.ex. om vi har ett kodord som är 7 byte långt så kan meddelandet i sig bestå av 5 symboler, medan den självrättande delen består av 2 stycken.

Exempel 18:
Säg att vi vill koda meddelandet 12345 som ett RS-kodord, vilket i binär form blir 001010011100101, då kan vi dela upp den på följande sätt:

[11] Sarah Spence Adams, s. 50.
[12] Tony Hill, s. 3.
[13] Van Lint, s. 99.
[14] Van Lint, s. 99.
[15] Hill, s. 10.

a_0x^4	a_1x^3	a_3x^2	a_2x^1	a_6x^0
001	010	011	100	101
1	2	3	4	5

Vårt meddelande kan då skrivas om till meddelandefunktionen:
$M(x) = 1x^4 + 2x^3 + 3x^2 + 4x + 5$ (här står siffrorna för element i $\mathbb{F}_8$).

Detta meddelande sätter vi till att ha 5 symboler för meddelandedelen och 2 stycken ytterligare för den självrättande delen. Koden i detta avslutande exempel kan identifiera och korrigera en felaktig symbol för varje kodord.

För att få fram de sista två vid slutet av kodordet behöver vi multiplicera vår sträng med x^2 för att flytta meddelandet två steg:

$$M(x) = 1x^6 + 2x^5 + 3x^4 + 4x^3 + 5x^2 + 0x + 0$$

De två avslutande självrättande symbolerna fås när vi utför polynomdivisionen $\frac{M(x)}{G(x)}$, där nämnaren är kodningspolynomet $G(x) = x^2 + 6x + 3$.

Innan vi utför divisionen repeterar vi hur addition och multiplikation går till i $\mathbb{F}_8$. Värt att komma ihåg är också att +/- är samma för dessa uträkningar.

Vill vi t.ex. ta 2+3 blir det genom bitvis operation av addition, eller XOR som den kallas i kodteori, 010_2 XOR $011_2 = 001_2 = 1$.

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	3	4	5	6	7	6
2	2	3	0	1	6	7	4	5
3	3	2	1	0	7	6	5	4
4	4	5	6	7	0	1	2	3
5	5	4	7	6	1	0	3	2
6	6	7	4	5	2	3	0	1
7	7	6	5	4	3	2	1	0

Tabell 4 – addition över $\mathbb{F}_8$

Om man skulle vilja ta exempelvis 3 x 5 blir det 011_2 x $101_2 = 100_2 = 4$. Detta fås genom:

		0	1	1	
	x	1	0	1	
		0	1	1	
	0	1	1		
0	1	1			
	1	1	1	1	XOR
	1	0	1	1	XOR mod 1011_2
		1	0	0	

Vi använder XOR mod 1011_2 (11_{10}) eftersom det irreducibla polynomet som används för att skapa kroppen är $x^3 + x + 1$.

x	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	3	1	7	5
3	0	3	6	5	7	4	1	2
4	0	4	3	7	6	2	5	1
5	0	5	1	4	2	7	3	6
6	0	6	7	1	5	3	2	4
7	0	7	5	2	1	6	4	3

Tabell 5 – multiplikation över $\mathbb{F}_8$

Låt oss nu ta reda på vad de två sista, självrättande symbolerna ska vara.

$$
\begin{array}{cccccccc|ccc}
 & & x^4 & 4x^3 & 5x^2 & 0 & x & & & & \\
\hline
x^6 & 2x^5 & 3x^4 & 4x^3 & 5x^2 & 0 & 0 & x^2 & 6x & 3 \\
x^6 & 6x^5 & 3x^4 & & & & & & & \\
\cline{1-3}
 & 4x^5 & 0 & 4x^3 & & & & & & \\
 & 4x^5 & 5x^4 & 7x^3 & & & & & & \\
\cline{2-4}
 & & 5x^4 & 3x^3 & 5x^2 & & & & & \\
 & & 5x^4 & 3x^3 & 4x^2 & & & & & \\
\cline{3-5}
 & & & & x^2 & 0 & 0 & & & \\
 & & & & x^2 & 6x & 3 & & & \\
\cline{5-7}
 & & & & & 6x & 3 & & &
\end{array}
$$

Därmed kommer vi lägga till 63 på slutet av vårt meddelande för att skapa kodordet:

$$C(x) = x^6 + 2x^5 + 3x^4 + 4x^3 + 5x^2 + 6x + 3$$

$$C(x) = 001\ 010\ 011\ 100\ 101\ 110\ 011$$

För att se om kodordet är giltigt kan vi dividera $C(x)$ med $G(x)$ och se att divisionen går jämnt ut, alltså att vi inte får någon rest.

Feldetektering

Hur gör vi då för att hitta och korrigera fel? Anta att mottagaren erhåller följande Felmeddelande (där rödmarkeringen avser felet):

$$R(x) = x^6 + 2x^5 + 3x^4 + 7x^3 + 5x^2 + 6x + 3$$

		x^4	$4x^3$	$5x^2$	0	0			
x^6	$2x^5$	$3x^4$	$7x^3$	$5x^2$	$6x$	3	1	6	3
x^6	$6x^5$	$3x^4$							
	$4x^5$	0	$7x^3$						
	$4x^5$	$5x^4$	$7x^3$						
		$5x^4$	0	$5x^2$					
		$5x^4$	$3x^3$	$4x^2$					
			$3x^3$	x^2	$6x$	0			
			$3x^3$	x^2	$6x$	3			
					$3x$	3			

Denna rest är unik, och för att mottagaren ska veta exakt hur meddelandet ska ändras kommer man utifrån denna tabell skapa en tabell som hjälper oss hitta det korrigerade meddelandet. För att få fram tabellen tar vi varje möjlig koefficient av varje exponentsvärde och dividerar med vår $G(x) = x^2 + 6x + 3$, som svarar mot 163, och tar det mod $x^3 + x + 1$ (1011_2).

÷163	x^0	x^1	x^2	x^3	x^4	x^5	x^6
1	1	10	100	1000	10000	100000	1000000
2	2	20	200	2000	20000	200000	2000000
3	3	30	300	3000	30000	300000	3000000
4	4	40	400	4000	40000	400000	4000000
5	5	50	500	5000	50000	500000	5000000
6	6	60	600	6000	60000	600000	6000000
7	7	70	700	7000	70000	700000	7000000

Tabell 6 – hjälptabell för att skapa vår tabell med rester

Dessa kommer man var och en ta och dividera med 163, till exempel för kolonn x^3 har vi 7000÷163:

		6x	4			
$7x^3$	0	0	0	x^2	$6x$	3
$7x^3$	$4x^2$	$2x$				
	$4x^2$	$2x$	0			
	$4x^2$	$5x$	7			
		7x	7			

och på så sätt skapa tabellen:

÷163	x^0	x^1	x^2	x^3	x^4	x^5	x^6
1	01	10	63	11	73	72	62
2	02	20	76	22	56	54	74
3	03	30	15	33	25	26	16
4	04	40	57	44	17	13	53
5	05	50	34	55	64	61	31
6	06	60	21	66	41	47	27
7	07	70	42	77	32	35	45

Tabell 7 – tabell med rester

I vårt exempel ovan fick vi resten 33, vilket i denna tabell ger oss en koefficient 3 för $x^{\mathbf{3}}$. Om vi sedan gör XOR på 7 och 3 i binär form får vi:

1	1	1	
0	1	1	XOR
1	0	0	

$100_2 = 4$ vilket då gör att det mottagna kodordet korrigeras till det avsedda kodordet när koefficienten för $x^{\mathbf{3}}$ ändras till 4 i stället för 7:

$$R(x) = x^6 + 2x^5 + 3x^4 + 4x^3 + 5x^2 + 6x + 3 = C(X).$$

Referenslista

Adams, Sarah Spence, *Introduction to Algebraic Coding Theory*, Cornell University, New York, USA, 2006. http://books.google.se/ (Hämtad 2021-09-25).

Biggs, Norman, *Discrete mathematics*, 2. ed., Oxford Univ. Press, Oxford, 2002

Cactus.io, http://cactus.io/tutorials/web/what-is-an-ascii-code (Hämtad 2021-09-25).

Castiñeira Moreira, Jorge. & Farrell, Patrick G., *Essentials of error-control coding*, John Wiley & Sons, West Sussex, England, 2006

Hill, Tony, *Reed Solomon Codes Explained*, https://www.slideshare.net/SimGrigoras/reed-solomon-explained-v1-0 (Hämtad 2021-09-25).

Lint, J. H. van, *Introduction to coding theory*, 3., rev. and expanded ed., Springer, Berlin, 1999

Trappe, Wade & Washington, Lawrence C., *Introduction to cryptography: with coding theory*, Prentice Hall, Upper Saddle River, NJ, 2002