Pro Gradu

Reedin ja Solomonin koodit

Katariina Huttunen

Jyväskylän yliopisto
Matematiikan laitos
Lokakuu 2016

Tiivistelmä

Huttunen Katariina, Reedin ja Solomonin koodit, matematiikan pro gradu-tutkielma, 49 s., Jyväskylän yliopisto, Matematiikan ja tilastotieteen laitos, syksy 2016.

Tutkielman tarkoituksena on esitellä Reedin ja Solomonin koodeja ja niiden ymmärtämiseksi tarvittavia esitietoja. Reedin ja Solomonin koodit ovat virheenkorjaamiskoodeja, joiden käsittelyssä käytetään äärellisiä kuntia. Virheenkorjaamiskoodeja tarvitaan kun dataa siirretään paikasta toiseen, koska siirron aikana voi tapahtua virheitä ja näin ollen perille tullut data eroaa alkuperäisestä. Virheenkorjaamiskoodien avulla alkuperäinen data voidaan mahdollisesti selvittää perille tulleesta viallisesta datasta. Edellä käytetään sanaa mahdollisesti, koska virheenkorjaamiskoodeilla on olemassa yläraja sille kuinka monta virhettä saa tapahtua, jotta alkuperäinen data voidaan vielä selvittää. Reedin ja Solomonin koodien kohdalla tämä yläraja riippuu koodin parametreistä n ja k yhtälön $t = \lfloor \frac{n-k+1}{2} \rfloor$ mukaan. Parametri n on koodin koodisanan pituus ja parametri k on koodin dimensio, toisin sanoen koodisanan varsinaista informaatiota sisältävän osan pituus. Varsinaisten informaatiota välittävän osan lisäksi virheenkorjaamiskoodeissa on pätkä dataa, jota käytetään virheenkorjaamiseen. Tähän osaan kuuluvia symboleita kutsutaan rendundanssisymboleiksi ja niiden määrä on siis $n - k$. Redundanssisymbolit ovat lineaarisesti riippuvia informaatiosymboleista.

Reedin ja Solomonin koodeista on olemassa syklinen versio ja alkuperäinen ei-syklinen versio. Sykliset Reedin ja Solomonin koodit ovat nykyisin enemmän käytetty muoto, koska niille on olemassa tehokkaita algoritmeja, joilla koodisanat voidaan purkaa alkuperäiseksi viestiksi. Syklisen Reedin ja Solomonin koodin koodisanojen pituus on $n = p^m - 1$ ja alkuperäisellä tavalla muodostetun Reedin ja Solomonin koodin koodisanan pituus on taas $n = p^m$. Tässä p^m on äärellisen kunnan $\mathbb{F}_{p^m}$ alkioiden lukumäärä. Äärellistä kuntaa käytetään Reedin ja Solomonin koodien molemmissa tapauksissa koodin aakkostona eli koodisanojen merkkisymbolit ovat jonkin sopivan äärellisen kunnan alkioita. Se mitä äärellistä kuntaa käytetään riippuu lähetettävän viestin koosta, halutusta virheenkorjaamiskyvystä ja minkälaista kanavaa käytetään.

Sisältö

1 Perusteita **3**
- 1.1 Koodi . . . 3
  - 1.1.1 Minimietäisyys ja koodin virheenkorjaamiskyky . . . 6
- 1.2 Äärelliset kunnat . . . 9
  - 1.2.1 Äärellisten kuntien muodostaminen Maximassa . . . 15
  - 1.2.2 Minimipolynomi . . . 17
  - 1.2.3 Äärellisen kunnan alkioilla laskeminen Maximassa . . . 21
- 1.3 Sykliset koodit . . . 23

2 Reedin ja Solomonin koodit **27**
- 2.1 Alkuperäinen Reedin ja Solomonin koodi . . . 27
- 2.2 Sykliset Reedin ja Solomonin koodit . . . 28
- 2.3 Yleistetyt Reedin ja Solomonin koodit . . . 33
- 2.4 Reedin ja Solomonin koodit ja virhekasaumat . . . 34

3 Viestin selvittäminen lähetetystä koodisanasta **36**
- 3.1 Viestin selvittäminen alkuperäisellä Reedin ja Solomonin koodilla koodatusta viestistä . . . 36
- 3.2 Petersonin, Gorensteinin ja Zierlerin koodinpurkaminen . . . 40
  - 3.2.1 Virhesyndrooman laskeminen . . . 40
  - 3.2.2 Virheenpaikannuspolynomin selvittäminen . . . 41
  - 3.2.3 Virheen paikan ja suuruuden selvittäminen . . . 43
- 3.3 Sudanin ja Guruswamin algoritmi . . . 45
- 3.4 Puuttuvan symbolin selvittäminen . . . 46

Johdanto

Nykymaailmassa liikkuu lukemattomat määrät dataa erilaisten kanavien välillä. Dataa siirtyy erilaisten tietokoneiden välillä mutta myös esimerkiksi kun cd-levyltä kuunnellaan musiikkia, niin cd-levylle tallennettu data siirtyy kanavan eli soittimen välityksellä eteenpäin ja tulee ilmi musiikkina. Kaikessa tässä datan siirtymisessä paikasta toiseen erilaisten kanavien kautta on mahdollista, että jossain vaiheessa siirtymistä tapahtuu jokin häiriö, jolloin siirtyvä data voi vioittua. Esimerkiksi, jos cd-levyyn tulee naarmu, niin levylle tallennettuun musiikkiin voi tulla häiriö. On kuitenkin olemassa keinoja, joilla näitä mahdollisia virheitä pystytään korjaamaan. Yksi näistä keinoista on virheenkorjauskoodit. Niiden avulla dataan voidaan liittää ominaisuus, joka pystyy tietyissä rajoissa korjaamaan dataan sattuneet virheet. Dataan lisätään tällöin pätkä lisää dataa. Se millä tavalla tämä pätkä lisätään riippuu koodin ominaisuuksista. Käytetään taas esimerkkinä cd-levyä. Nyt levyyn on tullut naarmu mutta koska levylle tallennettu data on ennen tallentamista koodattu virheenkorjaamiskoodin avulla, niin naarmun koosta ja paikasta riippuen, on mahdollista, että cd-levylle tallennetussa musiikissa ei havaita häiriötä sitä kuunneltaessa.

Tämän tutkielman tarkoituksena on esitellä Reedin ja Solomonin virheenkorjaamiskoodeja. Kyseessä on 1960 luvulla kehitetyt äärellisten kuntien rakennetta hyödyntävät koodit, joiden avulla voidaan korjata useita virheitä. Kyseiset koodit ovat erityisen tehokkaita virhekasaumien korjaamisessa eli tilanteissa, joissa virheitä tapahtuu monta peräkkäin. Reedin ja Solomonin koodeja käytetään esimerkiksi juuri cd-levyissä mutta ne ovat tärkeitä myös esimerkiksi avaruudessa tapahtuvassa viestinnässä.

Tutkielman ensimmäisessä luvussa käydään läpi mitä koodit ja virheenkorjaaminen ovat. Tämän jälkeen käsitellään äärellisiä kuntia ja niiden rakennetta niiltä osin kuin se on Reedin ja Solomonin koodien ymmärtämisen kannalta oleellista. Tästä siirrytään käsitellään syklisiä koodeja, koska ne ovat nykyisin yleisin Reedin ja Solomonin koodien käyttömuoto. Tutkielman toisessa luvussa esitellään varsinaisesti Reedin ja Solomonin koodit ja niiden erilaisia muotoja ja ominaisuuksia. Lisäksi tässä luvussa käydään läpi, kuinka viesti koodataan

Reedin ja Solomonin avulla sellaiseksi, että siihen tapahtuneita virheitä voidaan korjata. Kolmannessa luvussa käydään läpi kuinka selvittää vioittuneesta Reedin ja Solominin koodin avulla koodatusta viestistä alkuperäinen viesti. Neljännessä luvussa esitellään hieman Reedin ja Solomonin koodien käyttösovelluksia.

Tutkielman laskujen laskemisessa on hyödynnetty Maxima-tietokoneohjelmaa. Käytety ohjelmaversio on 5.31.2. Tutkielman liitteinä on maxima-koodeja, joilla esimerkkejä on laskettu. Suuret kiitokset Maximan käytön opetuksesta etenkin äärellisiä kuntia koskevien koodien kohdalla sekä muutoinkin pitkäjänteisestä ohjauksesta Ari Lehtoselle.

1 Perusteita

Tässä luvussa käydään läpi Reedin ja Solomonin koodien ymmärtämiseksi tarvittavia peruskäsitteitä ja joitakin niiden ominaisuuksia. Luvussa on käytetty pohjana MacWilliamsin ja Sloanen kirjaan The Theory of Error-Correcting codes (1977)[13] ja McEliecen kirjaa The Theory of Information and Coding (1985)[12].

1.1 Koodi

Koodeja on monenlaisia. Jotkut koodit ovat tarkoitettu viestin salaamista varten mutta jotkut koodit ovat taas virheenkorjaamista varten. Koodit voivat olla myös yhdistelmä näistä. Tässä tutkielmassa käsiteltävät koodit ovat nimenomaan virheenkorjauskoodeja.

Käydään aivan aluksi läpi mikä *koodi* C on. Koodit koostuvat pätkistä symbolijonoja, joita kutsutaan *koodisanoiksi* ja merkitään $c = (c_1, ..., c_n)$. Tämän tutkielman kannalta tärkeä asia on, että koodisanoja voidaan ilmaista polynomeina. Selkeyden vuoksi koodisanan $c = (c_1, ..., c_n)$ indeksointi kannattaa muuttaa muotoon $c = (c_0, c_1, ..., c_{n-1})$. Nyt koodisana voidaan ilmaista polynomina $c(x) = c_0 + c_1x + ... + c_{n-1}x^{n-1}$.

Koodisanojen symbolit tai koodisanapolynomien yhteydessä kertoimet ovat usein jonkin kunnan $\mathbb{F}_q$ alkioita. Kuntaa, josta symbolit ovat, kutsutaan koodin C *aakkostoksi*. Koodin koodisanojen ei ole välttämätöntä olla saman pituisia aina, mutta tässä tutkielmassa koodien kaikki koodisanat ovat sitä. Koodisanojen *pituutta* merkitään parametrillä n.

Virheenkorjauskoodien koodisanat sisältävät k kappaletta varsinaista viestiä koodaavaa symbolia eli *informaatiosymbolia*. Informaatiosymboleiden lisäksi koodisanoissa on vielä $n - k$ kappaletta *redundanssisymboleita*, joiden avulla koodi voi löytää ja korjata virheitä. Redundanssisymbolit ovat lineaarisesti riippuvia informaatiosymboleista.

Koodeja ilmaistaan monesti parametrien n ja k avulla muodossa $C(n, k)$. Annetaan seuraavaksi määritelmä koodille kuten se on määritelty McEliecen

kirjassa The Theory of Information and Coding [12, s. 140].

Määritelmä 1.1. *Lineaarinen koodi $C(n,k)$ kunnan $\mathbb{F}_q$ suhteen on n-ulotteisen vektoriavaruuden $V_n(\mathbb{F}_q) = \{(x_1, ..., x_n) | x_i \in \mathbb{F}_q\}$ k-ulotteinen aliavaruus.*

Aliavaruuden määritelmän nojalla lineaarisen koodin C koodisanoja voidaan siis laskea yhteen ja kertoa jollakin kertoimella $b \in \mathbb{F}_q$ siten, että syntyvä symboliketju on edelleen koodin C koodisana.

Lineaarisella koodilla $C(n,k)$ on vektoriavaruuden aliavaruutena olemassa kanta $g_1, ..., g_k$, missä $g_i \in C$ ovat koodisanoja. Kaikki koodin $C(n,k)$ muut koodisanat voidaan ilmaista näiden k koodisanojen lineaarikombinaationa $\sum_{i=1}^{k} b_i g_i$, missä $b_i \in \mathbb{F}_q$. Kun kantakoodisanoista muodostetaan matriisi asetamalla kukin kantakoodisana matriisin riviksi, niin saadaan $k \times n$ matriisi, jota kutsutaan *virittäjämatriisiksi* (generatormatrix) ja merkitään G.

Määritelmä 1.2. *Olkoon $C(n,k)$ lineaarinen koodi kunnan $\mathbb{F}_q$ suhteen. Matriisia G, jonka rivit muodostavat koodin C kannan, kutsutaan koodin C virittäjämatriisiksi.[10, s. 52].*

Virittäjämatriisin avulla voidaan antaa kompakti kuvaus koodille C. Ennenkaikkea matriisin G avulla voidaan helposti koodata viesti $m = (m_1, ..., m_k) \in V_k(\mathbb{F}_q)$ lineaarisen koodin C koodisanaksi $c = (c_1, ..., c_n)$:

$$m \to mG. \tag{1.1}$$

Kun koodattu viesti c lähetetään jonkin kanavan läpi, esimerkiksi avaruudesta maahan, niin kuten jo edellä todettiin, lähetyksessä voi tapahtua virheitä, jotka muuttavat koodisanaa c. Perille tulleesta viestistä $y = (y_1, ..., y_n) \in \mathbb{F}_q^n$ pitää pystyä jotenkin tarkastamaan, onko se oikein. Tämä voidaan tehdä käyttämällä hyväksi koodin C lineaarista komplementtia

$$C^{\perp} = \{r \in V_n(\mathbb{F}_q) | (c|r) = 0 \text{ kaikilla } c \in C\}. \tag{1.2}$$

Nyt sisätuloyhtälön

$$(y|r) = 0 \tag{1.3}$$

avulla voidaan testata, että onko saatu viesti y koodin C koodisana. Jos y toteuttaa yhtälön (1.3) kaikilla $r \in C^{\perp}$, niin kyseessä on koodin C koodisana. Yhtälöä (1.3) kutsutaan *pariteettiyhtälöksi*.

Koodin C lineaarinen komplementti $C^{\perp}$ on myös n-ulotteisen vektoriavaruuden $V_n(\mathbb{F}_q)$ aliavuruus ja sitä kutsutaan koodin C *duaalikoodiksi(dual code)*.

Duaalikoodin dimensio on koodin C lineaarisena komplementtina $n-k$. Duaalikoodille voidaan muodostaa myös virittäjämatriisi $n-k$ kantakoodisanan avulla kun nämä koodisanat asetetaan matriisin riveiksi. Syntyvää $(n-k)\times n$ matriisia kutsutaan koodin C *pariteetintarkistusmatriisiksi* ja sitä merkitään H. Annetaan pariteetintarkistusmatriisille vielä tarkka määritelmä.

Määritelmä 1.3. *Olkoon C(n,k) lineaarinen koodi kunnan $\mathbb{F}_q$ suhteen. Matriisi H, joka on koodin C duaalikoodin $C^\perp$ virittäjämatriisi, on koodin C pariteetintarkitusmatriisi. [10, s. 52]*

Lause 1.1. *Olkoon $C(n,k)$ lineaarinen koodi kunnan $\mathbb{F}_{p^m}$ suhteen ja olkoon G kyseisen koodin virittäjämatriisi. Tällöin vektorille $h \in \mathbb{F}_{p^m}$ pätee, että $h \in C^\perp$ jos ja vain jos $hG^T = 0$.*

Todistus.

1. Oletetaan ensin, että $h \in C^\perp$. Tällöin lineaarisen komplementin määritelmän nojalla $(c|h) = 0$ kaikilla $c \in C$. Erityisesti tämä pätee koodin C kantakoodisanoille $g_i, ..., g_k$, jotka muodostavat matriisin G eli pätee, että $hG^T = 0$.
2. Oletetaan sitten, että $hG^T = 0$. Koska matriisin G rivien avulla voidaan ilmaista kaikki koodin C koodisanat niin selvästi, jos $hG^T = 0$, niin myös $hc^T = 0$ kaikilla $c \in C$. Näin ollen $(c|h) = 0$ kaikilla $c \in C$ ja $h \in C^\perp$

□

Lause 1.2. *Olkoon $C(n,k)$ lineaarinen koodi kunnan $\mathbb{F}_{p^m}$ suhteen. Tällöin $(n-k)\times n$-matriisi H on koodin C pariteetintarkistusmatriisi, jos ja vain jos matriisin H rivit ovat lineaarisesti riippumattomat ja kaikille $c \in C$ pätee, että $Hc^T = 0$*

Todistus.

1. Oletetaan, että matriisi H on koodin C pariteetintarkistusmatriisi. Matriisi H on määritelmän nojalla koodin $C^\perp$ virittäjämatriisi, joten sen rivit muodostavat kyseisen koodin kannan ja siten ne ovat lineaarisesti riippumattomia. Edelleen kaikki kantakoodisanat ovat koodin $C^\perp$ koodisanoja, joten lauseen 1.1 nojalla $HG^T = 0$ eli myös $Hc^T = 0$ kaikilla $c \in C$.

2. Oletetaan, että $Hc^T = 0$ ja matriisin H rivit ovat riippumattomat. Tällöin lauseen 1.1 nojalla kaikki matriisin H rivit kuuluvat koodiin $C^\perp$. Koska matriisin H rivit ovat riippumattomat ja matriisin dimensio on $n-k$, niin se todellakin on koodin $C^\perp$ virittäjämatriisi ja näin ollen koodin C pariteetintarkistusmatriisi.

□

Nyt koodit on käyty lyhyesti läpi, joten siirrytään seuraavaksi minimietäisyyden tutkimiseen.

1.1.1 Minimietäisyys ja koodin virheenkorjaamiskyky

Yleensä koodin virheenkorjaamiskyvyn kannalta on tärkeätä, että koodisanat eroavat toisistaan tarpeeksi paljon. Yksi paljon käytetty koodien virheenkorjaustaktiikka on nimittäin korjata viallinen vektorijono y sitä lähinnä olevaksi koodisanaksi c (nearest-neighbor-coding). Lähimmän koodisanan määrittämiseksi koodisanoilla täytyy näin ollen olla jollakin tavalla määritelty etäisyys. Koodien kohdalla etäisyytenä käytetään usein *Hammingin etäisyyttä* ja se määritellään seuraavasti:

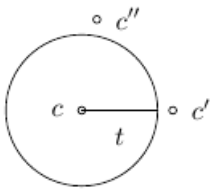
Määritelmä 1.4. *Hammingin etäisyys kahden vektorin $x = (x_1, ..., x_n)$ ja $y = (y_1, ..., y_n)$ välillä on niiden paikkojen lukumäärä, joissa x ja y eroavat toisistaan. Hammingin etäisyyttä merkitään $d_H(x, y)$. [13, s. 8]*

Määritelmä 1.5. *Olkoon C koodi, jossa on ainakin kaksi koodisanaa. Koodin C minimietäisyys $d_{min}(C)$ on pienin etäisyys erillisten koodisanojen välillä. Toisin sanoen siis $d_{min}(C) = \min\{d_H(c_i, c_j) | c_i, c_j \in C; c_i \neq c_j\}$. [12, s. 147]*

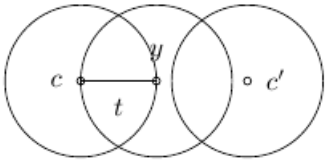
Oletetaan nyt, että lähetetään koodin C koodisana c mutta matkalla tapahtuukin t virhettä ja perille tulee vektori y. Sanotaan, että koodi C pystyy *tunnistamaan* tapahtuneet t virhettä, jos koodisana c ei muutu näiden t virheen sattuessa toiseksi koodin C kodisanaksi c'. Tällöin siis koodin C koodisanojen etäisyyden tulee olla enemmän kuin t. Tämä havainnollistetaan kuvassa 1.1. Jos koodi C on t-virheen tunnistava, niin se ilmoittaa yllä mainitussa tilanteessa, että lähetyksessä on tapahtunut virhe. Koodi ei kuitenkaan osaa ilmoittaa, että missä kohden vektoria y virheet sijaitsevat.

Edelleen koodi C pystyy *korjaamaan* t virhettä, jos se ensinnäkin pystyy tunnistamaan, että t virhettä on tapahtunut. Toisekseen koodin C pitää pystyä tunnistamaan kohdat, joissa virheet ovat tapahtuneet. Tätä varten viallisesta vektorista y pitää pystyä muuttamaan mitkä tahansa t symbolia siten, että muutosten seuraksena ei saada mitään muuta koodin C koodisanaa kuin koodisana c. Toisin sanoen kun muutetaan vektorin y oikeat vialliset t symbolia, niin saadaan koodisana c mutta muutoin muutoksien tuloksena on aina jokin muu vektori, joka ei ole koodin C koodisana. Kuvassa 1.2 havainnollistetaan tätä.

Kuva 1.1. Koodi C pystyy tunnistamaan t virhettä kun koodisanan c ympärille voidaan pirtää t-säteinen ympyrä ja mitään muita koodin C koodisanoja ei osu tämän ympyrän sisälle.



Kuva 1.2. Koodi C pystyy korjaamaan t virhettä kun virheellisen vektorin y ympärille voidaan pirtää t-säteinen ympyrä ja ainoa koodisana joka osuu tämän ympyrän sisälle on alkuperäinen koodisana c.



Huomautus. Jos käy niin, että kaksi koodisanaa ovat yhtä etäällä vektorista y ja ne ovat lähimmät koodisanat, niin tällöin koodi C ilmoittaa tasapelistä ja virhettä ei näin ollen voida korjata.

Kuvasta 1.2 voidaan jo päätellä, että koodin C koodisanojen pitää erota vähintään $2t+1$ kohdassa, jotta koodi C pystyy korjaamaan niihin sattuneet t virhettä. Tämä todistetaan seuraavassa lauseessa.

Lause 1.3. *Koodi C pystyy korjaamaan t virhettä jos ja vain jos $d_{min}(C) \geq 2t+1$.*

Todistus. 1. Oletetaan, että koodi C pystyy korjaamaan t virhettä. Jos $d_{min}(C) < 2t+1$, niin tällöin on olemassa kaksi koodisanaa c_i ja c_j siten, että $d(c_i, c_j) \leq 2t$. Huomataan, että jos $d(c_i, c_j) < t+1$, niin tällöin koodisana c_i voisi muuttua koodisanaksi c_j kun t symbolia muuttuu. Tämä on kuitenkin ristiriita, koska koodi C on t-virheen-korjaava. Täytyy siis olla $d(c_i, c_j) \geq t+1$. Nyt voidaan esimerkiksi olettaa, että koodisanat c_i ja c_j eroavat ensimmäisten d_m symbolien kohdalla. Nyt pätee, että $t+1 \leq d_m \leq 2t$. Olkoon nyt y saatu viesti. Voidaan kirjoittaa, että

$$y = y_1 \cdots y_t y_{t+1} \cdots y_{d_m} y_{d_m+1} \cdots y_n$$

missä symbolit $y_1, ..., y_t$ ovat samoja kuin koodisanassa c_i ja symbolit $y_{t+1}, ..., y_{d_m}$ ovat samoja kuin koodisanassa c_j sekä symbolit $y_{d_m+1}, ..., y_n$ ovat samoja kuin kummankin koodisanan lopussa. Nyt $d(c_i, y) = d_m - t \leq t = d(c_j, y)$. Tästä seuraa, että jos $d(c_i, y) < d(c_j, y)$, niin saatu viesti y tulkitaan koodisanaksi c_i, mikä on ristiriita oletuksen kanssa, koska tässä tilanteessa koodisanassa on tapahtunut vähemmän kuin t virhettä. Jos taas $d(c_i, y) = d(c_j, y)$ eli koodisanat ovat yhtä kaukana toisistaan, niin koodi ilmoittaa tasapelistä, mikä on taas ristiriita. Täytyy siis olla, että $d_{min}(C) \geq 2t + 1$.

2. Olkoon $d_{min}(C) \geq 2t + 1$. Olkoon c_i lähetetty koodisana ja y läpitullut vektori. Olkoon lisäksi $d(c_i, y) = t$. Nyt pätee, että jollekin toiselle koodisanalle c_j, $j \neq i$ on $d(c_j, y) \geq d(c_j, c_i) - d(c_i, y) \geq 2t+1-t = t+1 > d(c_i, y)$ eli kun vektori y tulkitaan lähimmäksi koodisanaksi, niin tuloksena on c_i. Tämä on oikea alkuperäinen koodisana, joten koodi kykenee korjaamaan t virhettä.

□

Koska minimietäisyyden avulla saadaan tietoa koodin virheenkorjaamiskyvystä, niin olisi hyvä, että minimietäisyys voitaisiin jollakin tehokkaalla tavalla määrittää. Isojen koodien kanssa ei ole tehokasta käydä kaikkia koodisanoja läpi ja laskea niiden etäisyyksiä. Seuraavan lauseen avulla saadaan yhteys koodin C pariteetintarkistusmatriisin ja minimietäisyyden välille.

Lause 1.4. *Olkoon H pariteetintarkistusmatriisi koodille, jonka pituus on n. Tällöin koodin minimietäisyys on d jos ja vain jo jokainen joukko $d-1$ matriisin H sarakkeita on lineaarisesti riippumattomia ja jotkin d saraketta ovat lineaarisesti riippuvia.*

Todistus. 1. Olkoon koodin minimi etäisyys $d < n$. Tällöin on olemassa koodisana $c = (c_1, ..., c_n)$, jolle $d_H(0, c) = d$ eli koodisanassa on d nollasta eroavaa symbolia. Koska kyseessä on koodisana, niin pariteettimatriisille H pätee, että $Hc^T = 0$ eli

$$Hc^T = 0 \leftrightarrow \begin{pmatrix} h_{11} & \cdots & h_{1n} \\ \vdots & \ddots & \vdots \\ h_{(n-k)1} & \cdots & h_{(n-k)n} \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} 0_1 \\ \vdots \\ 0_{n-k} \end{pmatrix}$$

Saadaan siis yhtälöryhmät

$$h_{11}c_1 + h_{12}c_2 + ... + h_{1n}c_n = 0$$
$$\vdots$$
$$h_{(n-k)1}c_1 + h_{(n-k)2}c_2 + ... + h_{(n-k)n}c_n = 0$$

Kun nämä lasketaan allekkain yhteen niin saadaan yhtälö

$$(h_{11} + ... + h_{(n-k)1})c_1 + ... + (h_{1n} + ... + h_{(n-k)n})c_n = 0 \tag{1.4}$$

Huomataan, että $h_{11} + ... + h_{(n-k)1}$ on matriisin H ensimmäinen sarake ja vastaavasti $h_{1n} + ... + h_{(n-k)n}$ on viimeinen sarake. Merkitään näitä matriisin H sarakkeita $\mathbf{v}_i = h_{1i}, ..., h_{(n-k)i}$, missä $i = 1, ..., n$. Yhtälö (1.4) saa nyt muodon

$$\mathbf{v}_1c_1 + ... + \mathbf{v}_nc_n = 0$$

Nyt koska d kappaletta $c_i \neq 0$, niin d määrälle vastaavia vektoreita $\mathbf{v}_i$ täytyy päteä, että $\sum \mathbf{v}_i c_i = 0$ vaikka $c_i \neq 0$ eli ne ovat lineaarisesti riippuvia. Jos jokin näistä edellä valituista d vektoreista jätetään pois ja otetaan siis jokin $d-1$ kokoinen joukko kyseisiä vektoreita $\mathbf{v}_i$, niin niille pätee, että $\sum \mathbf{v}_i c_i = 0$ jos ja vain jos $c_i = 0$ kaikilla i eli ne ovat lineaarisesti riippumattomia.

2. Olkoon matriisin H jotkin d saraketta lineaarisesti riippuvia. Koska pariteetintarkistusmatriisille pätee $Hc^T = 0$, aina kun c on koodisana, niin olemassa koodisana c, siten, että $d_H(0, c) = d$. Koska mikä tahansa joukko $d-1$ matriisin H sarakkeita on lineaarisesti riippumattomia, niin d on pienin mahdollinen etäisyys 0-vektorin ja c välillä. Täten d on koodin minimi etäisyys.

□

Siirrytään seuraavaksi koodisanojen etäisyydestä koodisanojen symboleihin. Toisin sanoen siirrytään käsittelemään äärellisiä kuntia, joita käytetään monien koodien, etenkin Reedin ja Solomonin koodien, aakkostona.

1.2 Äärelliset kunnat

Algebran peruskurssilta tuttu äärellinen kunta on $\mathbb{F}_p$, missä p on alkuluku. Näiden lisäksi äärellisiä kuntia ovat myös muotoa $\mathbb{F}_{p^m}$ olevat kunnat, missä p

on edelleen alkuluku ja $m > 1$. Lukua p kutsutaan kunnan *karakteristikaksi*. Aiemmin tutkielmassa koodin määritelmän kohdalla on esiintynyt kunta $\mathbb{F}_q$. Tämä kunta on juurikin äärellinen muotoa $\mathbb{F}_{p^m}$ oleva kunta.

Äärellisen kunnan $\mathbb{F}_{p^m}$ muodostamisessa tarvitaan *pääpolynomia* $\pi(x)$, jonka aste on m ja kertoimet ovat kunnasta $\mathbb{F}_p$. Pääpolynomin johtava kerroin on 1. Polynomin $\pi(x)$ tulee lisäksi olla *jaoton* eli se ei saa olla kahden tai useamman alempi asteisen vakiosta eroavan $\mathbb{F}_p$-kertoimisen polynomin tulo. Nyt kaikki polynomit, joiden aste on enintään $m-1$ ja kertoimet ovat kunnasta $\mathbb{F}_p$, muodostavat äärellisen kunnan $\mathbb{F}_{p^m}$, kun laskutoimitukset lasketaan modulo $\pi(x)$. Toisin sanoen $\mathbb{F}_{p^m} = \mathbb{F}_p[x]/\langle\pi(x)\rangle$.

Huomioidaan, että on mahdollista löytää useita jaottomia m-asteisia pääpolynomeja $\pi_i(x) \in \mathbb{F}_p[x]$. Koska vaaditut ehdot täyttyvät, niin kaikkien näiden avulla voidaan luoda äärellinen kunta $\mathbb{F}_{p^m}$. Voidaan kuitenkin osoittaa, että kaikki nämä syntyvät kunnat ovat isomorfisia keskenään. Toisin sanoen syntyvät kunnat ovat vain alkioiden esitysmuotoja vaille samat kunnat.

Lause 1.5. *Kaikki äärelliset kunnat, joissa on p^m alkiota, ovat isomorfisia keskenään.*

Edellisen lauseen todistusta ei esitellä tässä tutkielmassa, koska todistukseen tarvittaisiin lauseita, jotka eivät ole tämän tutkielman kannalta merkittäviä. Yksi versio todistuksesta löytyy Bhattacharyan, Jain ja Nagpaulin kirjasta Basic Abstract algebra. [3, s. 310-312]

Vielä ei ole osoitettu, että edellä kerrotulla tavalla tosiaan saaadan aikaan kunta. Näytetään seuraavassa lauseessa, että $\mathbb{F}_{p^m}$ täyttää kunnan kriteerit:

Lause 1.6. *Olkoon $\pi(x) \in \mathbb{F}_p[x]$ jaoton pääpolynomi. Tällöin $\mathbb{F}_{p^m} = \mathbb{F}_p[x]/\langle\pi(x)\rangle$ on kunta, jossa on p^m alkiota.*

Todistus. 1. Alkioiden määrä on todella p^m:
Kunnan alkiot voidaan esittää sivuluokkina $a = [a_0+a_1x+...+a_{m-1}x^{m-1}]_{\pi(x)}$, missä $a_i \in \mathbb{F}_p$. Näitä on p^m erilaista.

2. Kunnan kriteerit täyttyvät:

   (a) Joukosta $\mathbb{F}_{p^m}$ löydetään nolla-alkio $[0]_{\pi(x)}$, koska polynomi $p(x) = 0$ kuuluu alle $(m-1)$-asteisten polynomien joukkoon.

   (b) Joukosta $\mathbb{F}_{p^m}$ löydetään ykkösalkio $[1]_{\pi(x)}$, koska polynomi $p(x) = 1$ kuuluu alle $(m-1)$-asteisten polynomien joukkoon

(c) Joukossa $\mathbb{F}_{p^m}$ jokaisella alkioilla on olemassa vasta-alkiot eli $[a]_{\pi(x)} + [b]_{\pi(x)} = [0]_{\pi(x)}$ kaikilla $a, b \in \mathbb{F}_{p^m}$. Mistä tahansa alle $(m-1)$-asteisesta polynomista $p_a(x)$ saadaan polynomi $p(x) = 0$ kun siihen lisätään polynomi $p_b(x)$, jonka aste on sama ja kertoimet ovat polynomin $p_a(x)$ kertoimien vasta-alkiot. Tällaiset kertoimet löytyvät aina koska kertoimet tulevat kunnasta $\mathbb{F}_p$.

(d) Joukossa $\mathbb{F}_{p^m}$ jokaisella alkiolla on käänteisalkio eli $[a]_{\pi(x)}[b]_{\pi(x)} = [1]_{\pi(x)}$. Tämän näkemiseen tarvitaan seuraavaa lausetta.

□

Lause 1.7. *Jos $\pi(x) \in \mathbb{F}_p[x]$ on jaoton päääpolynomi ja $\deg \pi(x) = m$, niin jokaisella enintään $(m-1)$-asteisella polynomilla $B(x) \in \mathbb{F}_p[x]$ on yksikäsitteinen käänteispolynomi siten, että $B(x)B(x)^{-1} \equiv 1 \bmod \pi(x)$.*

Todistus. Tutkitaan tuloja $A_i(x)B(x) \bmod \pi(x)$. Polynomille $B(x)$ pätee, että $\deg B(x) \leq m-1$ ja sille lasketaan tulo kaikkien polynomien $A_i(x) \in \mathbb{F}_p[x]$ kanssa, joiden aste on enintään $m-1$. Kaikki nämä tulot ovat erisuuruisia.

Jos nimittäin olisi, että mille tahansa kahdelle eri polynomille $A_1(x)$ ja $A_2(x)$ pätee

$$A_1(x)B(x) = A_2(x)B(x) \bmod \pi(x) \Leftrightarrow (A_1(x) - A_2(x))B(x) = 0 \bmod \pi(x)$$

niin $\pi(x)|(A_1(x) - A_2(x))B(x)$. Tämä taas on mahdollista vain jos pätee, että $\deg((A_1(x) - A_2(x))B(x)) = lm$, jollekin $l \in \mathbb{N}$. Nyt kuitenkin koska $\pi(x)$ jaottomana polynomina ei ole kahden alempiasteisen polynomin tulo, niin tilanne $\pi(x)|(A_1(x) - A_2(x))B(x)$ on mahdollinen vain jos $\pi(x)|(A_1(x) - A_2(x))$ tai $\pi(x)|B(x)$. Koska $A_1(x) - A_2(x)$ ja $B(x)$ ovat enintään $m-1$ asteisia ja kuitenkin $\deg \pi(x) = m$, niin $\pi(x) \nmid (A_1(x) - A_2(x))$ ja $\pi \nmid B(x)$. Nyt siis $(A_1(x) - A_2(x))B(x) = 0 \bmod \pi(x)$ on mahdollinen vain jos $A_1(x) = A_2(x)$. Oletuksen mukaan $A_1(x)$ ja $A_2(x)$ ovat eri polynomeja, joten saadaan ristiriita. Näin saadaan, että kaikki tulot $A(x)B(x) \bmod \pi(x)$ ovat erisuuruisia.

Tulo $A_i(x)B(x) \bmod \pi(x)$ vastaa aina jotain enintään $(m-1)$-asteista polynomia, jonka kertoimet ovat kunnasta $\mathbb{F}_p$. Nyt koska kaikki tuloista $A_i(x)B(x) \bmod \pi(x)$, $i = 1, 2, ..$ ovat erisuuruisia ja polynomit $A_i(x) \in \mathbb{F}_p[x]$ käyvät läpi kaikki alle m olevat asteet, niin täten saadaan vastaavuus kaikkien $\mathbb{F}_p$-kertoimisten enintään $(m-1)$-asteisten polynomien ja tulojen $A_i(x)B(x) \bmod \pi(x)$ välille. Tällöin erityisesti jollekin $A_i(x)$ pätee, että $A_i(x)B(x) \equiv 1 \bmod \pi(x)$.

□

Äärellisen kunnan alkioita voidaan ilmaista m-pituisina pätkinä kunnan $\mathbb{F}_p$ alkioita. Kutsutaan tätä muotoa alkion listamuodoksi. Toinen tapa ilmaista alkioita on jäännösluokkapolynomin avulla. Esimerkiksi kunnan $\mathbb{F}_{3^2}$ alkiota 11 vastaa jäännösluokkapolynomi $[x+1]_{2+x+x^2}$ ja alkiota 20 vastaa jäännösluokkapolynomi $[2x]_{2+x+x^2}$. Merkintöjen yksinkertaistamiseksi sovitaan, että tästä edespäin alkion polynomimuotoa ilmaistaan vain jäännösluokan edustajan avulla kuten taulukossa 1.1 on tehty. Kyseisestä taulukosta nähdään juuri edellä mainittujen listamuotojen ja polynomimuotojen vastaavuus.

Alkion ilmaisutavan muuttaminen listamuodon ja polynomimuodon välillä ei ole yksikäsitteistä vaan alkiota 20 voisi vastata myös polynomi 2. Tämän kanssa täytyy olla tarkkana kuinka asiasta on sovittu kussakin yhteydessä. Tässä tutkielmassa äärellisen kunnan alkion polynomimuodosta saadaan listamuoto siten, että polynomin $m-1$. asteen kerroin on listamuodon ensimmäinen symboli ja niin edelleen kunnes tullaan vakiotermiin, joka vastaa aina listamuodon viimeistä symbolia.

Seuraavana määritellään *primitiivisen alkion* käsite. Primitiivinen alkio on tärkeä äärellisten kuntien ominaisuus, jolla on myös iso rooli Reedin ja Solomonin koodeja käsiteltäessä.

Määritelmä 1.6. *Alkio $\alpha \in \mathbb{F}_{p^m}$ on kunnan $\mathbb{F}_{p^m}$ primitiivinen alkio, jos sen välillä $[1, p^m - 1]$ olevien potenssien avulla voidaan ilmaista kaikki kunnan $\mathbb{F}_{p^m}$ nollasta eroavat alkiot. Sanotaan, että primitiivinen alkio virittää kunnan $\mathbb{F}_{p^m}$. [17, s. 9]*

Primitiivisen alkion avulla saadaan kolmas esitysmuoto kunnan $\mathbb{F}_{p^m}$ alkioille. Lisäksi primitiivisen alkion avulla äärellisen kunnan alkioiden välisistä kertolaskuista saadaan yksinkertaisempia.

Esimerkki 1.1. Olkoon α kunnan $\mathbb{F}_{p^m}$ primitiivinen alkio ja olkoon $\alpha^i, \alpha^j \in \mathbb{F}_{p^m}$ mielivaltaisia alkiota. Nyt, jos $i + j \leq p^m - 1$, niin

$$\alpha^i \alpha^j = \alpha^{i+j} = \alpha^k,$$

missä $\alpha^k \in \mathbb{F}_{p^m}$ on tutusti jokin korkeampi potenssinen alkio. Jos taas $i + j = l > p^m - 1$, niin

$$\alpha^i \alpha^j = \alpha^{i+j} = \alpha^l \overset{\star}{=} \alpha^{n(p^m-1)+r} = \alpha^{(p^m-1)^n} \alpha^r \overset{\star\star}{\equiv} \alpha^r,$$

missä $\alpha^r \in \mathbb{F}_{p^m}$.
$\star$ $l - n(p^m - 1) = r \leq p^m - 1$, missä kerroin $n \in \mathbb{N}$. Tällöin $r \in [1, p^m - 1]$.
$\star\star$ Fermat'n pieni lause 1.9, jolloin saadaan, että $\alpha^{p^m-1} \equiv 1 \mod (p^m - 1)$

Alkioiden kertolakussa syntyvät potenssien summat voidaan siis laskea mod $(p^m - 1)$ ja saadaan alkioiden tuloa vastaava kunnan $\mathbb{F}_{p^m}$ alkio.

Esimerkki 1.2. Määrätään kaikille kunnan $\mathbb{F}_{3^2}$ alkioille niiden kaikki kolme esitysmuotoa. Kunnan $\mathbb{F}_{3^2}$ muodostamisessa on käytetty polynomia $\pi(x) = 2 + x + x^2$ eli $\mathbb{F}_{3^2} = \mathbb{F}_3/\langle 2 + x + x^2 \rangle$. Olkoon α kunnan primitiivinen alkio. Alkion polynomimuodon ja primitiivisen alkion potenssin yhdistäminen tehdään seuraavasti:
Asetetaan $\pi(\alpha) = \pi([x]_{\pi(x)}) = 0$, jolloin saadaan

$$2 + \alpha + \alpha^2 = 0 \leftrightarrow \alpha^2 = -\alpha - 2 \equiv 2\alpha + 1 \mod 3.$$

Lasketaan sitten kaikille tätä korkeammille potensseille polynomimuoto:

$$\alpha^3 = \alpha^2\alpha = (1 + 2\alpha)\alpha = \alpha + 2\alpha^2 = \alpha + 2(1 + 2\alpha) = \alpha + 2 + 4\alpha$$
$$\equiv 2\alpha + 2 \mod 3$$
$$\alpha^4 = \alpha^2\alpha^2 = 1 + 4\alpha + 4\alpha^2 \equiv 1 + \alpha + \alpha^2 = 1 + \alpha + 1 + 2\alpha \equiv 2 \mod 3$$
$$\alpha^5 = \alpha^4\alpha \equiv 2\alpha \mod 3$$
$$\alpha^6 = \alpha^5\alpha = 2\alpha^2 = 2(1 + 2\alpha) \equiv \alpha + 2 \mod 3$$
$$\alpha^7 = \alpha^6\alpha = \alpha^2 + 2\alpha = 4\alpha + 1 \equiv \alpha + 1 \mod 3$$

Alkioiden listamuodot saadaan polynomimuodoista eli

$$\alpha \leftrightarrow 10$$
$$2\alpha + 1 \leftrightarrow 21$$
$$2\alpha + 2 \leftrightarrow 22$$
$$2 \leftrightarrow 02$$
$$2\alpha \leftrightarrow 20$$
$$\alpha + 2 \leftrightarrow 12$$
$$\alpha + 1 \leftrightarrow 11$$

Taulukossa 1.1 on vielä lueteltu kunnan $\mathbb{F}_{3^2}$ kaikki alkiot eri muodoissaan.

Määritelmä 1.7. *Alkion $a \in \mathbb{F}_{p^m}$ kertaluku* $\mathrm{ord}(a)$ *on pienin positiivinen kokonaisluku r, $1 \leq r \leq p^m - 1$, siten, että $a^r \equiv 1 \mod p^m$.*

Kunnan $\mathbb{F}_{p^m}$ alkion a kertaluku on alkion a eri suuruisten potenssien lukumäärä. Toisin sanoen niiden potenssien lukumäärä, joille pätee, että $a^j \neq a^k$, kun $0 < j < k < p^m$ [1, s. 88].

Taulukko 1.1. Kunnan $\mathbb{F}_{3^2}$ alkiot

00	0	0
10	α	α
21	$2\alpha + 1$	α^2
22	$2\alpha + 2$	α^3
02	2	α^4
20	2α	α^5
12	$\alpha + 2$	α^6
11	$\alpha + 1$	α^7
01	1	α^8

Primiitivinen alkio määriteltiin edellä siten, että sen eri potenssien avulla voidaan ilmaista kaikki kunnan $\mathbb{F}_{p^m}$ nollasta eroavat alkiot. Näin ollen primitiivisellä alkiolla tulee olla $p^m - 1$ eri potenssia, joten sen kertaluku on $p^m - 1$.

Lause 1.8. *Jokainen äärellinen kunta $\mathbb{F}_{p^m}$ sisältää primitiivisen alkion, jonka kertaluku on $p^m - 1$ ja jonka potenssien avulla kaikki kunnan alkiot voidaan ilmaista.*

Todistus. Olkoon n kaikkien nollasta eroavien kunnan $\mathbb{F}_{p^m}$ alkioiden kertalukujen maksimi. Olkoon nyt $\alpha \in \mathbb{F}_{p^m}$ se alkio, jonka kertaluku on n. Tällöin jokainen α^i, $1 \leq i \leq n$ tuottaa jonkin kunnan $\mathbb{F}_{p^m}$ alkion, joten $n \leq p^m - 1$. Olkoon nyt $\beta \in \mathbb{F}_{p^m}$ mielivaltainen nollasta eroava alkio, jonka kertaluku on d.

Oletetaan ensin, että $d \nmid n$. Tällöin on olemassa alkuluku p ja kokonaisluku k siten, että $p^k \mid d$ mutta $p^k \nmid n$. Annetaan nyt luvuille n ja d esitysmuodot $n = n'p^v$ ja $d = d'p^z$, missä n' ja d' eivät ole jaollisia luvulla p. Koska valittiin, että $p^k \mid d$, niin pätee, että $p^v < p^k \leq p^z$ eli $v < z$. Olkoon $\alpha' = \alpha^{p^v}$ ja $\beta' = \beta^{d'}$. Nyt $(\alpha')^{n'} = (\alpha^{p^v})^{n'} = \alpha^{n'p^v} = \alpha^n \equiv 1 \mod p^m - 1$ ja luku n' on selvästi pienin luku, jolle tämä pätee. Saadaan siis $\operatorname{ord}(\alpha') = n'$. Vastaavalla tavalla nähdään, että $\operatorname{ord}(\beta') = p^z$. Koska $p \nmid n'$, niin $\operatorname{syt}(\operatorname{ord}(\alpha'), \operatorname{ord}(\beta')) = 1$. Tästä seuraa, että $\operatorname{ord}(\alpha', \beta') = \operatorname{ord}(\alpha') \operatorname{ord}(\beta') = n'p^z$ [3, s. 79]. Nyt alun määritelmän mukaan luvun n tulisi olla suurin kunnan $\mathbb{F}_{p^m}$ alkioiden kertaluvuista mutta kuitenkin, koska kyseessä on äärellinen kunta, niin pätee, että $\alpha^{n'p^z} \in \mathbb{F}_{p^m}$ ja $\operatorname{ord}(\alpha'\beta') = n'p^z > n'p^v = n$. Tämä on ristiriita. Täytyy siis olla, että $d \mid n$. Näin ollen jokainen nollasta eroava kunnan $\mathbb{F}_{p^m}$ alkio β on polynomin $x^n - 1$ juuri, koska $\beta^n - 1 = (\beta^d)^{\frac{n}{d}} - 1 = 1 - 1 = 0$. Lisäksi koska polynomin $x^n - 1$ aste on n, niin sillä voi olla vain n juurta kunnassa $\mathbb{F}_{p^m}$, joten saadaan, että

$p^m - 1 \leq n$. Näin ollen saadaan, että $n = p^m - 1$. Nyt alkion α kaikki potenssit tuottavat eri kunnan alkion ja koska potensseja on $p^m - 1$, niin potenssien avulla voidaan ilmaista kaikki kunnan nollasta eroavat alkiot. Kyseessä on siis primitiivinen alkio. □

Huomautus. Primitiivinen alkio ei ole yksikäsitteinen. Tämä täytyy ottaa huomioon kun ilmaistaan jonkin kunnan alkioita primitiivisen alkion avulla.

Esimerkki 1.3. Tutkitaan kuntaa $\mathbb{F}_{3^2}$. Tässä kunnassa alkiolle x pätee, että $\text{ord}(x) = 8$ mutta myös alkiolle $x + 1$ pätee, että $\text{ord}(x + 1) = 8$. Näin ollen molemmat alkiot ovat kunnan $\mathbb{F}_{3^2}$ primitiivisiä alkioita.

Taulukossa 1.1 alkioiden potenssimuodot ja polynomimuodot on liitetty toisiinsa kun primitiivinen alkio $\alpha = x$. Vaihdetaan nyt, että primitiivinen alkio on $\alpha = x + 1$, jolloin taulukko on seuraavanlainen:

Taulukko 1.2. Kunnan $\mathbb{F}_{3^2}$ alkiot kun primitiivinen alkio on $\alpha = x + 1$

00	0	0
11	$x + 1$	α
12	$x + 2$	α^2
20	$2x$	α^3
02	2	α^4
22	$2x + 2$	α^5
21	$2x + 1$	α^6
10	x	α^7
01	1	α^8

Primitiivinen alkion valinta siis vaikuttaa siihen kuinka kunnan alkioita ilmaistaan.

1.2.1 Äärellisten kuntien muodostaminen Maximassa

Tässä kappaleessa esitellään kuinka Maximassa muodostetaan äärellisiä kuntia ja kuinka kunnan alkiot saadaan näkymään eri muodoissaan. Taulukossa 1.1 esitetyt kunnan $\mathbb{F}_{3^2}$ alkioiden eri muodot on laskettu Maximan avulla.

Äärellinen kunta luodaan Maximassa käyttämällä käskyä **gf_set_data()**. Äärellinen kunta $\mathbb{F}_{p^m}$ voidaan luoda kahdella tapaa. Kunnan voi luoda antamalla Maximalle parametrit p ja m, jolloin Maxima käyttää ohjelman sisälle määriteltyä m-asteista jaotonta polynomia. Toinen tapa luoda kunta on antaa kunnan karakteristika p ja jaoton polynomi, jonka avulla kunta halutaan

luoda. Nämä kaksi tapaa näkyvät alla olevassa Maximan koodipätkässä. Kun kunta on luotu Maximassa, niin sen tiedot saa esille käskyllä **gf_info()**, kuten myös alla näytetään. Jos luo monta erilaista kuntaa, niin täytyy olla tarkkana, koska käsky **gf_info()** antaa aina viimeisimmäksi luodun kunnan tiedot näkyviin.

```
(%i1)  gf_set_data(3,2);
```

(%o1) $Structure[GF - DATA]$

```
(%i1)  gf_set_data(3,2);
```

(%o1) $Structure[GF - DATA]$

```
(%i2)  gf_info();
```

$characteristic = 3$
$reductionpolynomial = x^2 + 1$
$primitiveelement = x + 1$
$nrofelements = 9$
$nrofunits = 8$
$nrofprimitiveelements = 4$

(%o2) $false$

```
(%i3)  gf_set_data(3,x^2+x+2);
```

(%o3) $Structure[GF - DATA]$

```
(%i4)  gf_info();
```

$characteristic = 3$
$reductionpolynomial = x^2 + x + 2$
$primitiveelement = x$
$nrofelements = 9$
$nrofunits = 8$
$nrofprimitiveelements = 4$

(%o4) $false$

Kunnan $\mathbb{F}_{p^m}$ primitiivinen alkio saadaan näkyviin myös käskyllä **gf_primitive()**. Primitiivisen alkion kertaluku saadaan käskyllä **gf_order()**. Nämä käskyt näkyvät alla olevassa Maximan koodipätkässä. Kyseisessä pätkässä kunnan luomisessa on käytetty polynomia $\pi(x) = x^2 + x + 2$.

```
(%i5)  gf_primitive();
```
(%o5) x

```
(%i6)  gf_order();
```
(%o6) 8

Jatketaan edelleen kunnalla $\mathbb{F}_{p^m}$, joka on luotu polynomin $\pi(x) = x^2 + x + 2$ avulla. Seuraavaksi nähdään kuinka Maximassa saadaan kunnan nollasta eroaville alkioille muodot primitiivisen alkion potenssina, polynomina ja listana.

```
(%i7)  potenssimuoto:makelist(gf_primitive()^i,i,1,gf_order());
```
(%o7) $[x, x^2, x^3, x^4, x^5, x^6, x^7, x^8]$

```
(%i8)  polynomeina:map(gf_eval, potenssimuoto);
```
(%o8) $[x, 2\,x + 1, 2\,x + 2, 2, 2\,x, x + 2, x + 1, 1]$

```
(%i9)  listamuoto:map(gf_p2l, polynomeina);
```
(%o9) $[[1, 0], [2, 1], [2, 2], [2], [2, 0], [1, 2], [1, 1], [1]]$

Käsky **gf_eval()** sieventää polynomeja. Se laskee yli $m - 1$ astetta olevat polynomit modulo $\pi(x)$, missä $\pi(x)$ on siis jaoton polynomi. Sen avulla voidaan helposti sieventää polynomit siten, että lopputulokseksi saadaan jonkin kunnan $\mathbb{F}_{p^m}$ alkion polynomimuoto. Miinus-merkkiset potenssimuodot saadaan myös helposti ilmaistua positiivisina potensseina kun käytetään apuna käskyä **gf_eval()**. Luvussa 1.2.3 nähdään kuinka edellinen temppu onnistuu. Käsky **gf_p2l()** muuttaa polynomimuodon listamuodoksi. Päinvastainen käsky on **gf_l2p()**.

1.2.2 Minimipolynomi

Minimipolynomin avulla äärelliselle kunnalle $\mathbb{F}_{p^m}$ saadaan tärkeä rakenteellinen ominaisuus. Tässä kappaleessa esitellään joitakin tärkeitä minimipolynomin ominaisuuksia.

Määritelmä 1.8. *Alkion $\alpha^i \in \mathbb{F}_{p^m}$ minimipolynomi kunnan $\mathbb{F}_p$ suhteen on alinta astetta oleva pääpolynomi $M_i(x)$ siten, että $M_i(\alpha^i) = 0$. [13, s. 99]*

Lause 1.9. *(Fermat'n pieni lause) Jokainen kunnan $\mathbb{F}_{p^m}$ alkio toteuttaa yhtälön*

$$x^{p^m} = x. \tag{1.5}$$

Toisin sanoen jokainen kunnnan $\mathbb{F}_{p^m}$ alkio on yhtälön $x^{p^m} = x$ juuri.

Todistus. Kaikki kunnan $\mathbb{F}_{p^m}$ nollasta eroavat alkiot voidaan ilmaista primitiivisen alkion avulla ja primitiivisen alkion kertaluku on $p^m - 1$. Tästä seuraa, että kaikki nollasta eroavat alkiot toteuttavat yhtälön $x^{p^m-1}-1 = 0$. Myös alkio 0 toteuttaa yhtälön $0^{p^m} = 0$. Näin ollen kaikki kunnan $\mathbb{F}_{p^m}$ alkiot toteuttavat yhtälön

$$x(x^{p^m-1} - 1) = x^{p^m} - x = 0 \leftrightarrow x^{p^m} = x.$$

□

Lause 1.10. *Kunnan $\mathbb{F}_{p^m}$ alkion α^i minimipolynomi $M_i(x)$ kunnan $\mathbb{F}_p$ suhteen on aina olemassa ja yksikäsitteinen. Tämä minimipolynomi on myös jaoton kunnassa $\mathbb{F}_p$.*

Todistus.

1. Olemassaolo: Fermat'n lauseen mukaan jokainen kunnan $\mathbb{F}_{p^m}$ alkio toteuttaa yhtälön $x^{p^m} - x = 0$, joten tämä takaa, että minimipolynomi on olemassa.

2. Yksikäsitteisyys: Antiteesi: Olkoon $M_1(x)$ ja $M_2(x)$ kaksi alkion $\alpha^i \in \mathbb{F}_{p^m}$ minimipolynomia. Jakoyhtälön mukaan saadaan

$$M_1(x) = k(x)M_2(x) + r(x), \tag{1.6}$$

missä $\deg r(x) < \deg M_2(x)$ tai $r(x) = 0$. Nyt $M_1(\alpha) = 0 = M_2(\alpha)$, joten yhtälöstä (1.6) seuraa, että $r(\alpha) = 0$. Nyt ei voi olla, että $r(\alpha) = 0$ ja $\deg r(x) < \deg M_2(x)$, joten täytyy olla, että $r(x) = 0$. Tällöin $M_2(x)|M_1(x)$. Toisaalta jakoyhtälö voidaan ilmoittaa myös muodossa

$$M_2(x) = l(x)M_1(x) + r(x),$$

jolloin saadan vastaavasti, että $M_1(x)|M_2(x)$. Täten täytyy olla, että $M_1(x) = M_2(x)$.

3. Jaottomuus kunnassa $\mathbb{F}_p$: Antiteesi: Olkoon minimipolynomi $M_i(x)$ jaollinen kunnassa $\mathbb{F}_p$. Tällöin $M_i(x) = g(x)f(x)$, missä $\deg g(x), \deg f(x) < \deg M_i(x)$ ja $f(x) \in \mathbb{F}_p$ ja $g(x) \in \mathbb{F}_p$ ovat pääpolynomeja. Koska $0 =$

$M_i(\alpha^i) = g(\alpha^i)f(\alpha^i)$, niin täytyy olla joko $f(\alpha^i) = 0$ tai $g(\alpha^i) = 0$. Tämä on ristiriita, koska $M_i(x)$ on minimipolynomi. Siispä $M_i(x)$ on jaoton kunnassa $\mathbb{F}_p$.

□

Lause 1.11. *Olkoon $M_i(x) \in \mathbb{F}_p[x]$ alkion $\alpha^i \in \mathbb{F}_{p^m}$ minimipolynomi. Tällöin $M_i(x)|x^{p^m} - x$.*

Todistus. Jokainen kunnan $\mathbb{F}_{p^m}$ alkio toteuttaa Fermat'n lauseen mukaan yhtälön $p(x) = x^{p^m} - x = 0$. Täytyy olla $\deg p(x) \geq \deg M(x)$, jolloin jakoyhtälöä hyödyntämällä saadaan, että $p(x) = k(x)M_i(x) + r(x)$, missä $r(x) = 0$ tai $\deg r(x) < M_i(x)$. Nyt $0 = p(\alpha) = k(\alpha)M_i(\alpha) + r(\alpha)$, joten minimipolynomin määritelmän nojalla täytyy olla, että $r(x) = 0$. Tällöin $M_i(x)|x^{p^m} - x$. □

Lause 1.12. *Olkoon $\alpha_i \in \mathbb{F}_{p^m}$. Tällöin $(\sum_{i=1}^n \alpha_i)^p = \sum_{i=1}^n \alpha_i^p$, kun $n \geq 2$.*

Todistus. Induktio:

1. $n = 2$: Binomikaavalla saadaan, että $(\alpha_1 + \alpha_2)^p = \sum_{k=0}^p \binom{p}{k} \alpha_1^{p-k}\alpha_2^k$, missä $\binom{p}{0} = 1 = \binom{p}{p}$. Huomataan lisäksi, että $\binom{p}{k} = \frac{p(p-1)\cdots(p-k+1)}{k!} = 0 \mod p$. Saadaan siis, että $(\alpha_1 + \alpha_2)^p = \alpha_1^p + \alpha_2^p$.
2. induktio-oletus: väite pätee, kun $n = k$
3. $n = k+1$: $(\sum_{i=1}^{k+1} \alpha_i)^p \overset{\star}{=} (\sum_{n=1}^k \alpha_i)^p + \alpha_{k+1}^p \overset{\star\star}{=} \sum_{n=1}^k \alpha_i^p + \alpha_{k+1}^p = \sum_{n=1}^{k+1} \alpha_i^p$.
   $\star$ kohta 1.
   $\star\star$ induktio-oletus

□

Lause 1.13. *Kunnan $\mathbb{F}_{p^m}$ alkioilla α ja α^p on sama minimipolynomi.*

Todistus. Olkoon nyt $M \in \mathbb{F}_p[x]$ alkion $\alpha \in \mathbb{F}_{p^m}$ minimipolynomi. Tällöin määritelmän nojalla pätee, että $M(\alpha^i) = \sum_{j=0}^s m_j \alpha^j = 0$, missä $m_j \in \mathbb{F}_p$. Nyt

$$M(\alpha^p) = \sum_{j=0}^{s} m_j(\alpha^p)^j \overset{\star}{=} \sum_{j=0}^{s} m_j^p(\alpha^j)^p = \sum_{j=0}^{s}(m_j\alpha^j)^p \overset{\star\star}{=} (\sum_{j=0}^{s} m_j\alpha^j)^p = (M(\alpha))^p = 0 \tag{1.7}$$

joten alkion alkion α minimipolynomi on myös alkion α^p minimipolynomi.
$\star$ Fermat'n pienen lauseen nojalla $m_j = m_j^p$.
$\star\star$ Lause 1.12 □

Edellistä lausetta hyödyntämällä nähdään, että myös niillä kunnan $\mathbb{F}_{p^m}$ alkioilla α^i, jotka ovat muotoa $\alpha^i = \alpha^{p^j}$, $j = 1, 2, ..$ on sama minimipolynomi kuin alkiolla α. Näiden alkioiden potenssit muodostavat joukon $\{p, p^2, ..., p^{k-1}\}$, missä k on pienin kokonaisluku siten, että $p^k \equiv 1 \mod p^m - 1$. Merkitään tätä joukkoa C_1.

Otetaan sitten kunnan $\mathbb{F}_{p^m}$ alkio $\alpha^s \notin C_1$, $s \in \mathbb{N}$. Myös tälle alkiolle voidaan käyttää edellistä lausetta, jolloin siis alkioilla α^s ja $(\alpha^s)^p = \alpha^{sp}$ on sama minimipolynomi. Edelleen alkioilla α^{sp^j} on sama minimipolynomi. Näiden alkioiden potenssit muodostavat joukon $\{s, sp, sp^2, ..., sp^{k-1}\}$, missä $sp^k \equiv s \mod p^m - 1$. Merkitään tätä joukkoa C_s.

Tätä kunnan alkioiden jakamista minimipolynomien avulla voidaan jatkaa kunnes kaikki kunnan $\mathbb{F}_{p^m}$ alkioit kuuluvat johonkin joukkoon C_s, missä $s \in \mathbb{N}$.

Minimipolynomit tosiaan jakavat kunnan $\mathbb{F}_{p^m}$ alkiot erillisiin joukkoihin. Näitä joukkoja kutsutaan *syklotomisiksi sivuluokiksi*. Edelleen alkioita, joiden potenssit kuuluvat samaan syklotomiseen sivuluokkaan, kutsutaan *konjugaattialkioiksi*.

Määritelmä 1.9. *Kokonaisluvun s sisältävä syklotominen sivuluokka modulo $p^m - 1$ on joukko $C_s = \{s, ps, p^2s, ..., p^{k-1}s\}$, missä k on pienin positiivinen kokonaisluku siten, että $p^k s \equiv s \mod p^m - 1$. [13, s. 104]*

Lause 1.14. *1. Minimipolynomille pätee, että $M_i(x) = \prod_{i \in C_s}(x - \alpha^i)$.*

2. Polynomille $x^{p^m-1} - 1$ pätee, että $x^{p^m-1} - 1 = \prod_s M_s(x)$, missä s käy läpi syklotomisten sivuluokkien modulo $p^m - 1$ edustajat.

Todistus. 1. Olkoon $M_i(x) \in \mathbb{F}_p$ alkion $\alpha^i \in \mathbb{F}_{p^m}$ minimipolynomi. Kaikille alkion α^i konjugaattialkioille α^j, missä $j \in C_s$, pätee myös $M_i(\alpha^j) = 0$. Näin ollen minimipolynomin tulee olla ainakin muotoa

$$M_i(x) = \prod_{i \in C_s} (x - \alpha^i). \tag{1.8}$$

Määritelmän mukaan alkion $\alpha^i \in \mathbb{F}_{p^m}$ minimipolynomi on alinta astetta oleva pääpolynomi. Näin ollen minimipolynomi on yhtälössä (1.8) olevaa muotoa, koska tämä on alinta mahdollista astetta oleva pääpolynomi, jolle $M_i(\alpha^j) = 0$ kaikilla konjugaattialkioilla α^j.

2. Fermat'n lauseen nojalla $x^{p^m} - x = \prod_{\beta \in \mathbb{F}_{p^m}}(x - \beta)$. Tästä ja kohdasta 1. saadaan $x^{p^m-1} - 1 = \prod_s M_s(x)$, missä s käy läpi syklotomisten sivuluokkien modulo $p^m - 1$ edustajat.

□

Esimerkki 1.4. Halutaan löytää kunnan $\mathbb{F}_{3^2} = \mathbb{F}_3/\langle 2+x+x^2\rangle$ kaikien alkioiden minimipolynomit $M_{\alpha_i}(x) \in \mathbb{F}_3$. Olkoon α kunnan primitiivinen alkio, jolloin sille pätee $\text{ord}(\alpha) = 8$. Ensimmäiseksi täytyy selvittää siis mitkä alkiot ovat keskenään konjugaattialkioita eli mitkä alkiot kuuluvat samaan syklotomiseen sivuluokkaan. Lasketaan nämä sivuluokat. Laskut suoritetaan modulo 8:

$$
\begin{aligned}
&C_0 = \{0\} \\
&C_1 : 1, 1 \times 3 \equiv 3, 1 \times 3^2 \equiv 1, \text{ eli } C_1 = \{1, 3\} \\
&C_2 : 2, 2 \times 3 \equiv 6, 2 \times 3^2 \equiv 2, \text{ eli } C_2 = \{2, 6\} \\
&C_3 = C_1 \\
&C_4 : 4, 4 \times 3 \equiv 4, \text{ eli } C_4 = \{4\} \\
&C_5 : 5, 5 \times 3 \equiv 7, 5 \times 3^2 \equiv, \text{ eli } C_5 = \{5, 7\} \\
&C_6 = C_2 \\
&C_7 = C_5
\end{aligned}
$$

Kun syklotomiset sivuluokat ovat selvillä, niin voidaan laskea minimipolynomit. Näitä on tässä tapauksessa 5 erilaista. Minimipolynomit on nimetty kunkin syklotomisen sivuluokan ensimmäisen alkion mukaan.

$$
\begin{aligned}
&M_0(x) = x - 1 \\
&M_1(x) = (x - \alpha)(x - \alpha^3) = x^2 - (\alpha^3 + \alpha)x + \alpha^4 = x^2 - \alpha^4 x + \alpha^4 = x^2 + x + 2 \\
&M_2(x) = (x - \alpha^2)(x - \alpha^6) = x^2 + 1 \\
&M_4(x) = x - \alpha^4 = x + 1 \\
&M_5(x) = (x - \alpha^5)(x - \alpha^7) = x^2 - x + \alpha^4 = x^2 + \alpha^5 x + 2 = x^2 + 2\alpha x + 2
\end{aligned}
$$

Näin on saatu selville kunnan $\mathbb{F}_{3^2}$ minimipolynomit.

1.2.3 Äärellisen kunnan alkioilla laskeminen Maximassa

Näytetään seuraavaksi kuinka Maximassa voidaan laskea äärellisen kunnan alkioilla. Esimerkissä 1.4 on käytetty hyväksi Maximaa.

Maximassa luodaan kunta $\mathbb{F}_{3^2} = \mathbb{F}_3/\langle 2 + x + x^2\rangle$. Lasketaan nyt esimerkkinä alkioiden α ja α^3 minimipolynomi. Esimerkin 1.4 ensimmäinen vaihe on laskea kertolasku auki. Tämä on tehty manuaalisesti. Tämän jälkeen polynomia

on alettu sieventämään ja se on tehty Maximan avulla. Alla olevassa koodipätkässä on ensin asetettu summa $\alpha^3 + \alpha$ ja tämän jälkeen se on sievennetty käskyn **gf_eval()** avulla polynomiksi 2. Tälle polynomille on lasketty logaritmi eli vastaava primitiivisen alkion potenssi käskyllä **gf_log()**, jolloin on saatu, että kyseessä on α^4. Seuraavaksi on laskettu tämän alkion vasta-alkio. Käsky **gf_eval()** palauttaa aina polynomin, joten sen avulla saadulle polynomille 1 on laskettu vastaava primitiivisen alkion potenssi. Näin on siis saatu, että $-(\alpha^3 + \alpha) = -\alpha^4 = 1$

```
(%i1)  gf_set_data(3, x^2+x+2);
```

(%o1) $Structure[GF - DATA]$

```
(%i2)  a:gf_primitive()^3+gf_primitive();
```

(%o2) $x^3 + x$

```
(%i3)  b:gf_eval(a);
```

(%o3) 2

```
(%i4)  gf_log(b);
```

(%o4) 4

```
(%i5)  gf_eval(-gf_primitive()^4);
```

(%o5) 1

```
(%i6)  gf_log(1);
```

(%o6) 0

Muut minimipolynomit voidaan laskea vastaavalla tavalla. Toinen ja nopeampi tapa laskea alkion minimipolynomi on käskyllä **gf_minimal_poly()** kuten alla on tehyt.

```
(%i1)  gf_set_data(3, x^2+x+2);
```

(%o1) $Structure[GF - DATA]$

```
(%i2)  gf_minimal_poly(x);
```

(%o2) $z^2 + z + 2$

```
(%i3)  gf_minimal_poly(2*x+2);
```

(%o3) $z^2 + z + 2$

1.3 Sykliset koodit

Reedin ja Solomonin koodit ovat usein *syklisiä koodeja*. Syklisen koodin koodisanoille pätee, että kun koodisanan viimeinen symboli siirretään koodisanan ensimmäiseksi symboliksi, ts. tehdään *syklinen siirto*, niin näin syntyvä vektori on myös koodisana. Se ei ole kuitenkaan sama koodisana kuin aiempi. McEliecen kirjassa The Theory of Information and Coding syklinen koodi määritellään seuraavasti: [12, s. 167]

Määritelmä 1.10. *Lineaarinen koodi C(n,k) kunnan $\mathbb{F}_{p^m}$ suhteen on syklinen, jos jokaiselle koodisanalle $c = (c_0, ..., c_{n-1})$ pätee, että $c^R = (c_{n-1}, c_0, ..., c_{n-2}) \in C$.*

Lause 1.15. *Olkoon koodi $C(n, k)$ on syklinen koodi kunnan $\mathbb{F}_{p^m}$ suhteen. Tällöin pätee, että $c^R(x) = xc(x) \bmod (x^n - 1)$.*

Todistus. Olkoon koodin C koodisana $c(x) = c_0 + c_1x + ... + c_{n-1}x^{n-1}$. Kun tälle tehdään syklinen siirto, niin saadaan

$$c_{n-1} + c_0x + c_1x^2 + ... + c_{n-2}x^{n-1} = c_{n-1} + xc(x) - c_{n-1}x^n$$
$$= xc(x) + c_{n-1}(1 - x^n) = xc(x) - c_{n-1}(x^n - 1)$$

Tästä saadaan, että

$$c_{n-1} + c_0x + c_1x^2 + ... + c_{n-2}x^{n-1} \equiv xc(x) \bmod (x^n - 1)$$

Eli kun tehdään syklinen siirto koodisanalle $c(x)$, niin saadaan koodisana $xc(x) \bmod (x^n - 1)$. □

Koska koodin C koodisanoille voidaan tehdä syklinen siirto, joka tuottaa toisen saman koodin koodisanan kun lasketaan modulo $x^n - 1$, niin sykliset koodit kuuluvat jäännösluokkarenkaaseen $R_n = \mathbb{F}_{p^m}[x]/(x^n - 1)$.

Määritelmä 1.11. *Jos C on syklinen koodi, niin alinta astetta olevaa koodin C pääpolynomia(=koodisanaa) kutsutaan koodin virittäjäpolynomiksi. Tätä virittäjäpolynomia merkitään $g(x)$.[12, s. 173]*

Koodi C määritellään monesti virittäjäpolynomin avulla. Seuraava lause antaa tärkeitä virittäjäpolynomin ominaisuuksia.

Lause 1.16. *Olkoon koodi $C(n,k)$ syklinen ja $g(x)$ koodin C virittäjäpolynomi. Tällöin*

1. *$g(x)$ on yksikäsitteinen,*
2. *$C = \langle g(x) \rangle$,*
3. *$g(x)|(x^n - 1)$,*
4. *jos $\deg g(x) = n - k$, niin koodin C dimensio on k ja sen kanta on $\{g(x), xg(x), ..., x^{k-1}g(x)\}$.*
5. *jos $\deg g(x) = n - k$, niin jokainen koodin C (n,k) koodisana voidaan ilmaista yksikäsitteisesti muodossa $g(x)f(x)$, missä $f(x) = 0$ tai $deg f(x) < k$ ja $f(x) \in \mathbb{F}_{p^m}[x]$*

Todistus.

1. Olkoon $r(x)$ toinen alinta astetta oleva pääpolynomi C:ssä. Tällöin siis $\deg g(x) = \deg r(x) = s$. Aliavaruutena C on suljettu yhteen- ja vähennyslaskun suhteen. Nyt $g(x) - r(x) \in C$ ja $\deg(g(x) - r(x)) < s$. Tämä on ristiriita, koska koodiin C kuuluvan koodisanapolynomin alin mahdollinen aste piti olla s. Näin ollen $g(x)$ on yksikäsitteinen.

2. Olkoon $c(x) \in C$ mielivaltainen. Koska $g(x)$ on alinta astetta oleva koodisanapolynomi koodissa C, niin sen ja jakoyhtälön avulla koodisana $c(x)$ voidaan esittää muodossa $c(x) = g(x)f(x) + h(x)$, missä $f(x), h(x) \in \mathbb{F}_{p^m}[x]$. Lisäksi $h(x) = 0$ tai $\deg h(x) < \deg g(x)$. Ehdosta $c(x), g(x) \in C$ seuraa, että myös $h(x) \in C$, koska koodi C on aliavaruus. Koska kuitenkin $g(x)$ on alinta astetta oleva koodisanapolynomi koodissa C, niin ehdosta $\deg h(x) < \deg g(x)$ seuraa että, $h(x) = 0$. Näin ollen, koska $c(x)$ on mielivaltainen koodin C koodisanapolynomi ja se on muotoa $c(x) = g(x)f(x)$, niin mikä tahansa koodin C koodisana voidaan ilmaista tässtä muodossa. Toisin sanoen $g(x)$ todella virittää koodin C.

3. Esitetään polynomi $x^n - 1$ jakoyhtälön avulla, jolloin $x^n - 1 = g(x)f(x) + k(x)$, missä taas $f(x), k(x) \in \mathbb{F}_{p^m}[x]$ ja lisäksi $k(x) = 0$ tai $\deg k(x) < \deg g(x)$. Huomataan, että jäännösluokkarenkaassa $\mathbb{F}_{p^m}[x]/\langle x^n - 1 \rangle$ polynomi $x^n - 1$ vastaa nollapolynomia. Tällöin se vastaa nollapolynomia myös ideaalissa eli koodissa C. Toisin sanoen $x^n - 1$ vastaa koodissa C koodisanaa 0. Tässä tärkentä on, että $x^n - 1$ vastaa koodin C koodisanaa.

Tällöin saadaan samoin kuin kohdassa 2., että $k(x) = 0$. Tästä seuraa, että $g(x)|x^n - 1$.

4. & 5. Kohdan 2. nojalla jokainen koodisana voidaan esittää muodossa $c(x) = g(x)f(x)$, missä $f(x) \in \mathbb{F}_{p^m}[x]$. Nyt jos $c(x) = 0$, niin täytyy olla, että $f(x) = 0$, koska $g(x) \neq 0$ aina. Muistetaan, että $c(x) = c_0 + c_1x + ... + c_{n-1}x^{n-1}$ eli $\deg c(x) = n-1$. Jos $c(x) \neq 0$, niin edellisestä muistutuksesta ja oletuksesta $\deg g(x) = n - k$ seuraa, että $\deg f(x) = k - 1 < k$. Näin ollen saadaan, että $C = \{g(x)f(x) | f(x) = 0 \text{ tai } \deg f(x) < k\}$. Edelleen joukko $\{g(x), xg(x), ..., x^{k-1}g(x)\}$ virittää koodin C ja sen dimensio on k.

□

Edellisessä lauseessa nähdään, että virittäjäpolynomille $g(x)$ pätee, että $g(x)|x^n - 1$. Tällöin on hyödyllistä osata jakaa polynomi $x^n - 1$ tekijöihin kunnassa $\mathbb{F}_{p^m}$. Kuinka tämä onnistuu?

Alkion $\alpha^i \in \mathbb{F}_{p^m}$ minimipolynomille pätee lauseen 1.14 nojalla, että $M_{\alpha^i}(x) = \prod_{i \in C_s}(x - \alpha^i)$. Edelleen pätee, että $x^{p^m-1} - 1 = \prod_s M_s(x)$, missä M_s on syklotomisen sivuluokan C_s alkioiden minimipolynomi ja luku $s \in \mathbb{N}$ käy läpi kaikki kunnan $\mathbb{F}_{p^m}$ syklotomisten sivuluokkien edustajat. Nyt yleistetään tätä lausetta siten, että käytetään *p-syklotomisia sivuluokkia modulo n kunnan $\mathbb{F}_{p^m}$ suhteen.* Nämä ovat myös muotoa $C_s = \{s, sp, sp^2, ..., sp^{k-1}\}$, missä $sp^k \equiv s \mod n$. Näin ollen kokonaisluvut $\{0, 1, ..., n-1\}$ jakautuvat p-syklotomisiin sivuluokkiin modulo n eli $\{0, 1, ..., n-1\} = \bigcup_s C_s$, missä s käy läpi p-syklotomisten sivuluokkien modulo n edustajat. Tästä saadaan, että

$$x^n - 1 = \prod_s M_s(x), \tag{1.9}$$

missä s käy läpi kunnan $\mathbb{F}_{p^m}$ suhteen olevien p-syklotomisten sivuluokkien modulo n edistajat ja M_s on p-syklotomisen sivuluokan C_s minimipolynomi.

Esimerkki 1.5. Halutaan löytää polynomin $x^8 - 1$ jaottomat tekijät kunnassa $\mathbb{F}_{3^2}$. Esimerkissä 1.4 laskettiin minimipolynomit kaikille alkioille. Tiedetään, että $x^n - 1 = \prod_s M_s(x)$ eli $x^8 - 1 = M_0(x)M_1(x)M_2(x)M_4(x)M_5(x)$. Näin ollen saadaan, että $x^8 - 1 = (x-1)(x^2+x+2)(x^2+1)(x+1)(x^2+2\alpha x+2)$.

Syklisen koodin virittäjäpolynomi $g(x)$ muodostuu siis joidenkin polynomin $x^n - 1$ tekijöiden avulla. Toisin sanoen

$$g(x) = \prod_{i \in T} (x - \alpha^i), \tag{1.10}$$

missä $T = \cup C_{s'}$ ja s' käy läpi jotkin p-syklotomisten sivuluokkien modulo n edustajista. Joukkoa T kutsutaan virittäjäpolynomin $g(x)$ määrittämän *koodin C määrittelyjoukoksi*. Alkioita $\alpha^i \in T$ kutsutaan *koodin nolliksi*, koska koodisanoille $c(x)$ pätee, että $c(\alpha^i) = 0$ kaikilla $i \in T$. Koodin määrittelyjoukolla on tärkeä rooli koodin virheenkorjaamiskykyä määrittäessä.

2 Reedin ja Solomonin koodit

Vuoden 1960 kesäkuussa julkaistiin Journal of the Society for Industrial and Applied Mathematics lehdessä Irvingin Reedin ja Gus Solomonin artikkeli nimeltä Polynomial Codes over Certain Finite Fields. Tässä artikkelissa esiteltiin virheenkorjauskoodi, jota nykyisin kutsutaan Reedin ja Solomonin koodiksi. Tämän koodin aakkostona on jokin äärellinen kunta $\mathbb{F}_{p^m}$. Monesti käytännön sovelluksissa data siirtyy binäärisessä muodossa paikasta toiseen ja täten monesti käytännössä Reedin ja Solomonin koodien aakkostona käytetty kunta on muotoa $\mathbb{F}_{2^m}$, missä $m \geq 1$. Reedin ja Solomonin koodeilla on monia käyttökohteita. Niitä käytetään avaruudessa tapahtuvassa datan siirrossa esimerkiksi satelliiteissa ja kaukaisen avaruuden tutkimisessa. Ne ovat tärkeitä myös maan päällä tapahtuvassa langattomassa viestinnässä. Myös datan tallentaminen esimerkiksi cd-levyille on yksi paljon käytetty Reedin ja Solomonin sovellus, mutta tämä ei ehkä enää nykypäivänä ole se tärkein sovellus, kun cd-levyjen käyttö on vähentynyt, koska esimerkiksi musiikki on siirtynyt enenevissä määrin internettiin.

Reedin ja Solmonin koodeja tutkittaessa törmää enimmäkseen kahteen tapaan koodata näitä koodeja. Toinen tapa on Irving Reedin ja Gus Solmonin alkuperäinen tapa ja toinen tapa on myöhemmin kehitetty virittäjäpolynomin avulla tapahtuva koodaaminen.

2.1 Alkuperäinen Reedin ja Solomonin koodi

Alkuperäisen Reedin ja Solomonin koodin idea on melko yksinkertainen. Siinä koodisanat muodostetaan seuraavasti:

1. Olkoon $m_0, .., m_{k-1}$ joukko viestisymboleita, missä $m_i \in \mathbb{F}_{p^m}$.
2. Muodostetaan viestisymboleista polynomi
   $P(x) = m_0 + m_1 x + ... + m_{k-2}x^{k-2} + m_{k-1}x^{k-1}$.

3. Lasketaan polynomin $P(x)$ arvo kaikilla alkioilla $\alpha^i \in \mathbb{F}_{p^m}$, jolloin saadaan koodisana $\mathbf{c} = (c_0, c_1, ..., c_{p^m-1}) = \{P(0), P(\alpha), ..., P(\alpha^{p^m-1})\}$.

4. Annetaan viestisymboleiden $m_0, .., m_{k-1}$ saada kaikkia mahdollisia arvoja kunnassa $\mathbb{F}_{p^m}$, jolloin kohtia 1-3 toistamalla saadaan loput koodin koodisanat.

Tällä tavoin muodostetun koodin koodisanojen pituus on $n = p^m$. Tällä tavoin muodostetussa Reedin ja Solomonin koodissa on p^{mk} koodisanaa. Koodi on myös selvästi lineaarinen, koska kahden $k-1$-asteisen polynomin $P_1(x)$ ja $P_2(x)$ summa on myös $k-1$-asteinen polynomi. Toisin sanoen kahden koodisanan mudostamiseen käytettävien polynomien $P_1(x)$ ja $P_2(x)$ avulla voidaan muodostaa polynomi $P_3(x)$, jonka avulla voidaan muodostaa kolmas koodisana.

2.2 Sykliset Reedin ja Solomonin koodit

Syklinen Reedin ja Solomonin koodi on nykyisin enemmän käytetty kuin alkuperäinen Reedin ja Solomonin koodin muoto. Syklisiä Reedin ja Solomonin koodeja käsiteltäessä on tärkeätä muistaa seuraavia aiemmin esiteltyjä käsitteitä: minimietäisyys (määritelmä 1.5), virittäjäpolynomi $g(x)$ ja sen lauseessa 1.16 esitellyt ominaisuudet, minimipolynomi $M(x)$ (määritelmä 1.8) sekä syklisen koodin määrittelyjoukko (s. 25).

Sykliset Reedin ja Solomonin koodit ovat BCH-koodeiksi kutsuttujen koodien alaluokka, joten määritellään ensin lyhyesti BCH-koodit.

Määritelmä 2.1. *Syklinen (n,k)-koodi C kunnan $\mathbb{F}_{p^m}$ suhteen on BCH-koodi, jonka suunniteltu etäisyys on δ, jos jollekin kokonaisluvulle $b \geq 0$ pätee*

$$g(x) = \text{pyj}\{M_b(x), M_{b+1}(x), ..., M_{b+\delta-2}(x)\}.$$

Toisin sanoen $g(x)$ on alinta astetta oleva pääpolynomi, jolle pätee

$$g(\alpha^b) = g(\alpha^{b+1}) ... = g(\alpha^{b+\delta-2}) = 0.$$

[13, s. 202]

Lause 2.1. *(BCH-bound) Olkoon $C(n,k)$ syklinen koodi kunnan $\mathbb{F}_{p^m}$ suhteen, jonka virittäjäpolynomille $g(x)$ pätee joillakin kokonaisluvuilla $b \geq 1$ ja $\delta \geq 1$, että*

$$g(\alpha^b) = g(\alpha^{b+1}) = ... = g(\alpha^{b+\delta-2}) = 0. \tag{2.1}$$

Toisin sanoen koodin määrittelyjoukossa T on vähintään $\delta - 1$ peräkkäistä alkiota. Tällöin koodin minimietäisyys on δ

Todistus. Olkoon $c = (c_0, c_1, .., c_{n-1})$ koodin C koodisana. Polynomina koodisana c on $c(x) = c_0 + c_1 x + ... + c_{n-1}x^{n-1}$. Muistetaan, että lauseen 1.16 nojalla jokainen koodin $C(n,k)$ koodisanan c voidaan ilmaista virittäjäpolynomin avulla eli $c(x) = g(x)f(x)$, missä $f(x) = 0$ tai $\deg f(x) < k$. Tällöin oletuksesta seuraa, että

$$c(\alpha^b) = c(\alpha^{b+1}) = ... = c(\alpha^{b+\delta-2}) = 0. \tag{2.2}$$

Tässä siis esimerkiksi

$$c(\alpha^b) = c_0 + c_1\alpha^b + c_2\alpha^{2b} + ... + c_{n-1}\alpha^{(n-1)b} = 0$$

Kirjoittamalla kaikki yhtälölöt (2.2) auki ne voidaan ilmaista myös matriisimuodossa

$$\begin{pmatrix} 1 & \alpha^b & \alpha^{2b} & \cdots & \alpha^{(n-1)b} \\ 1 & \alpha^{b+1} & \alpha^{2(b+1)} & \cdots & \alpha^{(n-1)(b+1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{b+\delta-2} & \alpha^{2(b+\delta-2)} & \cdots & \alpha^{(n-1)(b+\delta-2)} \end{pmatrix} \begin{pmatrix} c_0 \\ \vdots \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} 0_0 \\ \vdots \\ 0_{n-1} \end{pmatrix} \tag{2.3}$$

Oletetaan nyt, että koodin C minimietäisyys on pienempi kuin δ. Tällöin koodisanassa c on siis $w \leq \delta - 1$ nollasta eroavaa symbolia. Oletetaan, että koodisanassa c on tasan $w = \delta - 1$ nollasta eroavaa symbolia. Olkoon koodisanan nollasta eroavat symbolit paikoilla $\{c_{i_1}, ..., c_{i_w}\}$. Tällöin esimerkiksi saadaan, että

$$c(\alpha^b) = c_{i_1}\alpha^{i_1 b} + ... + c_{i_w}\alpha^{i_w b} = 0.$$

Nyt yhtälö (2.3) saadaan muotoon

$$\begin{pmatrix} \alpha^{i_1 b} & \alpha^{i_2 b} & \cdots & \alpha^{i_w b} \\ \alpha^{i_1(b+1)} & \alpha^{i_2(b+1)} & \cdots & \alpha^{i_w(b+1)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{i_1(b+\delta-2)} & \alpha^{i_2(b+\delta-2)} & \cdots & \alpha^{i_w(b+\delta-2)} \end{pmatrix} \begin{pmatrix} c_{i_1} \\ c_{i_2} \\ \vdots \\ c_{i_w} \end{pmatrix} = 0$$

Huomataan, että vasemman puoleinen matriisi on $(\delta - 1) \times w$- eli $w \times w$-matriisi ja, että vektorit $(\alpha^{i_1 b}, ..., \alpha^{i_w b}), ..., (\alpha^{i_1(b+\delta-2)}, ..., \alpha^{i_w(b+\delta-2)})$ ovat lineaarisesti riippuvia. Tällöin niiden muodostaman matriisin determinantti on nolla. Kuitenkin huomataan myös, että

$$\det \begin{pmatrix} \alpha^{i_1 b} & \cdots & \alpha^{i_w b} \\ \alpha^{i_1(b+1)} & \cdots & \alpha^{i_w(b+1)} \\ \vdots & \ddots & \vdots \\ \alpha^{i_1(b+\delta-2)} & \cdots & \alpha^{i_w(b+\delta-2)} \end{pmatrix} = \alpha^{(i_1+...+i_w)b} \det \begin{pmatrix} 1 & \cdots & 1 \\ \alpha^{i_1} & \cdots & \alpha^{i_w} \\ \vdots & \vdots & \vdots \\ \alpha^{i_1(\delta-2)} & \cdots & \alpha^{i_w(\delta-2)} \end{pmatrix}$$

missä oikean puoleisin matriisi on samoin kuin edellä $w \times w$-matriisi. Huomataan myös, että kyseessä on Vandermonden neliömatriisi, jolloin sen determinantti on Vandermonden determinanttina nollasta eroava [13, s. 116]. Kuitenkin myös $\alpha^{(i_1+...+i_w)b}$ on nollasta eroava. Tämä on ristiriita. Täytyy siis olla, että koodisanan c nollasta eroavien symbolien määrä $w \neq \delta - 1$. Jos $w = delta - i$, $i = 2, ..., \delta - 2$, niin saadaan vastaavalla tavalla ristiriidat, jolloin saadaan, että koodisanassa c täytyy olla enemmän kuin $\delta - 1$ nollasta eroavaa symbolia. Näin ollen koodin C minimietäisyys on vähintään δ. □

BCH-koodit ovat tärkeä syklisten koodien luokka, joilla on paljon käyttösovelluksia. BCH-boundin nojalla, BCH-koodeille pätee, että niille voidaan suunnitella minimietäisyys δ. Tämä on hyödyllinen ominaisuus virheenkorjaamista ajatellen, koska näin ollen voidaan määritellä kuinka monta virhettä koodi pystyy korjaamaan. Määritellään seuraavaksi Reedin ja Solomonin koodi.

Määritelmä 2.2. *Sykliset Reedin ja Solomonin koodit ovat BCH-koodeja, joille pätee, että $n = p^m - 1$. Niiden virittäjäpolynomi on muotoa*

$$g(x) = (x - \alpha^b)(x - \alpha^{b+1}) \cdots (x - \alpha^{b+\delta-2}) \tag{2.4}$$

missä δ on koodin suunniteltu etäisyys. [13, s. 294]

Yleensä Reedin ja Solomonin koodien tapauksessa $b = 1$.

Koska Reedin ja Solomonin koodin pituus on $n = p^m - 1$, niin $\mathrm{ord}_n(p^m) = 1$. Tällöin polynomin $x^n - 1$ kaikki juuret ovat kunnassa $\mathbb{F}_{p^m}$ ja ne ovat kaikki kunnan $\mathbb{F}_{p^m}$ nollasta eroavat alkiot. Edelleen kaikki syklotomiset sivuluokat

mod n sisältävät vain yhden alkion. Näin ollen on Reedin ja Solomonin koodin määrittelyjoukkoon on mahdollista valita juurikin $\delta - 1$ peräkkäistä alkiota. Reedin ja Solomonin koodien virheenkorjaamiskyky voidaan siis määritellä tarkasti.

Lauseen 1.16 nojalla syklisen Reedin ja Solomonin koodin koodisanat voidaan muodostaa kaavalla

$$c(x) = m(x)g(x). \tag{2.5}$$

Toinen tapa muodostaa Reedin ja Solomonin koodien koodisanat on laskea ne alla esitetyllä tavalla. Näiden kahden tavan ero on siinä, että alla olevan yhtälön (2.6) mukaisella tavalla muodostetussa koodisanassa viesti on näkyvissä suoraan koodisanasta. Näin ollen jos koodisana c lähetetään, niin sen jälkeen kun ollaan varmistuttu, että kyseessä on oikea c, niin viesti voidaan lukea suoraan koodisanasta kun tiedetään niiden koordinaatit. Nämä taas tiedetään kun tiedetään kuinka koodisana muodostetaan. Yhtälön (2.5) tavalla koodattu viesti saadaan näkyviin kun koodisana lasketaan modulo $g(x)$. Tapojen ero on siis siinä missä vaiheessa modulo laskutoimitus tehdään. Nyt

$$c(x) = x^{n-k}m(x) + (x^{n-k}m(x) \mod g(x)). \tag{2.6}$$

Yhtälössä (2.6) viestipolynomia siirretään hieman eteenpäin, jotta koodisanan alkuun saadaan tilaa virheenkorjaukseen tarvittaville redundanssisymboleille, jotka saadaan yhtälöstä $x^{n-k}m(x) \mod g(x)$.

Esimerkki 2.1. Halutaan lähettää viesti $m = (1, 2, 2, 0, 2, 1, 1, 1)$ ja halutaan, että viestistä tulee ainakin puolet oikein perille. Nyt voidaan valita aakkostoksi kunta $\mathbb{F}_{3^2}$, jonka muodostamisessa käytetään taas polynomia $x^2 + x + 2$. Näin saadaan, että viesti on muotoa $m = (\alpha^6, \alpha^5, \alpha^2, \alpha^7) = (m_0, m_1, m_2, m_3)$. Käytettävä koodi on näin ollen $RS(8, 4)$-koodi, joka voi korjata $t = 2$ virhettä. Kun viesti muutetaan polynomiksi saadaan $m(x) = \alpha^6 + \alpha^5 x + \alpha^2 x^2 + \alpha^7 x^3$. Koodin suunniteltu etäisyys on $\delta = 5$ ja määrittelyjoukoksi voidaan valita $T = \{1, 2, 3, 4\}$, jolloin virittäjäpolynomiksi saadaan

$$g(x) = (x - \alpha^1)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4) = \alpha^2 - x + \alpha^2 x^2 - \alpha^3 x^3 + x^4$$

missä $\alpha \in \mathbb{F}_{3^2}$ primitiivinen alkio. Koodisana voi olla nyt kahta muotoa:

1. $c(x) = m(x)g(x)$

2. $c(x) = x^4 m(x) + (x^4 m(x) \mod g(x))$.

Jos koodisana lasketaan tavalla 1., niin saadaan

$$c(x) = 1 + \alpha^4 x + \alpha x^2 + \alpha^5 x^3 + \alpha^4 x^4 + \alpha x^5 + \alpha^7 x^7.$$

Toisin sanoen koodisana on $c = (1, \alpha^4, \alpha, \alpha^5, \alpha^4, \alpha, 0, \alpha^7)$.

Huomautus. Tässä täytyy olla tarkkana, jotta ei sekoita koodisanan polynomimuodon muodostamista äärellisen kunnan alkioiden listamuodon muuttamiseen polynomimuotoon. Äärellisen kunnan alkioiden kohdalla vasemmanpuoleinen symboli on syntyvän polynomin korkeimman asteen kerroin mutta koodisanan kohdalla tämä menee juuri toisin päin ja vasemmanpuoleisin symboli on vakiotermi.

Tapa 2. vaatii hieman enemmän laskemista. Lasketaan ensin

$$x^4 m(x) = x^4(\alpha^6 + \alpha^5 x + \alpha^2 x^2 + \alpha^7 x^3) = \alpha^6 x^4 + \alpha^5 x^5 + \alpha^2 x^6 + \alpha^7 x^7$$

Seuraavaksi lasketaan $x^4 m(x) \mod g(x)$. Lasketaan tämä polynomien jakolasku esimerkin vuoksi jakokulmassa.

$x^4 m(x) \mod g(x)$:

$$\begin{array}{r|l} & \alpha^7 x^3 + \alpha^6 x^2 + \alpha^5 x + \alpha^5 \\ \hline x^4 - \alpha^3 x^3 + \alpha^2 x^2 - x + \alpha^2 & \alpha^7 x^7 + \alpha^2 x^6 + \alpha^5 x^5 + \alpha^6 x^4 \\ & \underline{\alpha^7 x^7 - \alpha^2 x^6 + \alpha x^5 - \alpha^7 x^4 + \alpha x^3} \\ & \alpha^6 x^6 + \alpha x^5 + \alpha^5 x^4 + \alpha^5 x^3 \\ & \underline{\alpha^6 x^6 - \alpha x^5 + \alpha^0 x^4 - \alpha^6 x^3 + \alpha^0 x^2} \\ & \alpha^5 x^5 + \alpha^3 x^4 + \alpha^4 x^3 + \alpha^4 x^2 \\ & \underline{\alpha^5 x^5 - \alpha^0 x^4 + \alpha^7 x^3 - \alpha^5 x^2 + \alpha^7 x} \\ & \alpha^5 x^4 + \alpha^2 x^3 + \alpha^3 x^2 + \alpha^3 x \\ & \underline{\alpha^5 x^4 - \alpha^0 x^3 + \alpha^7 x^2 - \alpha^5 x + \alpha^7} \\ & \alpha^3 x^3 + \alpha^7 x^2 + \alpha^6 x + \alpha^3 \end{array} \tag{2.7}$$

$x^4 m(x) \mod g(x) = \alpha^3 x^3 + \alpha^7 x^2 + \alpha^6 x + \alpha^3$

Tästä saadaan, että

$$c(x) = \alpha^3 + \alpha^6 x + \alpha^7 x^2 + \alpha^3 x^3 + \alpha^6 x^4 + \alpha^5 x^5 + \alpha^2 x^6 + \alpha^7 x^7$$

Toisin sanoen koodisana on

$$c = (\alpha^3, \alpha^6, \alpha^7, \alpha^3, \alpha^6, \alpha^5, \alpha^2, \alpha^7) = (2, 2, 1, 2, 1, 1, 2, 2, 1, 2, 2, 0, 2, 1, 1, 1)$$

Huomataan, että tavassa 2. alkuperäinen viesti tosiaan löytyy muuttumattomana koodisanasta.

Yksi syklisten Reedin ja Solomonin koodien tärkeä ominaisuus on se, että ne ovat MDS-koodeja eli maximum distance separable-koodeja. MDS-koodit ovat koodeja, joiden koodisanojen välinen pituus (distance) on suurin mahdollinen (maximum) kyseisillä koodi parametreillä. Toisin sanoen koodille pätee, että $d_{min} = n - k + 1$. Lisäksi näiden koodien koodisanoista voidaan erotella (separate) viestisymbolit ja pariteetintarkistussymbolit. Koodaus voidaan tehdä joko systemaattisesti tai epäsystemaattisesti. Systemaattisessa koodauksessa viesti on luettevissa koodisanasta suoraan tai laskemalla koodisana modulo $g(x)$. Epäsystemaattisesssa koodaauksessa viestin selvittämiseksi tarvitaan enemmän laskemista viestin selvittämiseksi. MDS-koodit ovat siis laskennallisesti käteviä jos ei ole tarvetta jostain syystä salata viestiä.

Lause 2.2. *(Singleton bound) Olkoon C (n, k)-koodi. Tällöin $n - k \geq d_{min} - 1$.*

Todistus. Nyt $r = n - k$ on pariteettimatriisin H aste. Tällöin r on matriisin H lineaarisesti riippumattomien sarakkeiden suurin mahdollinen määrä. Näin ollen Lauseen 2.2 avulla saadaan, että $d_{min} \leq n - k + 1$. □

Edellisen lauseen avulla voidaan osoittaa, että sykliset Reedin ja Solomonin koodit ovat MDS-koodeja.

Lause 2.3. *Sykliset Reedin ja Solomonin koodit ovat MDS.*

Todistus. Koska RS-koodi on syklinen (n, k)-koodi yli kunnan $\mathbb{F}_{p^m}$ ja koodin suunniteltu etäisyys on δ ja määrittelyjoukko $T = \{b, b + 1, ..., b + \delta - 2\}$, niin BCH-bound antaa $d_{min}(C) \geq \delta$. Singleton-bound taas antaa, että $k \leq n - d_{min} + 1$ eli $d_{min}(C) \leq n - k + 1$, kun $d_{min}(C) < n$. Eli saadaan $n - k + 1 \geq d_{min}(C) \geq \delta$. Jos δ on nyt RS-koodin suunniteltu etäisyys, niin $\delta = 2t + 1 = n - k + 1$, joten $d_{min}(C) = n - k + 1$. □

2.3 Yleistetyt Reedin ja Solomonin koodit

Esitellään tässä kappaleessa lyhyesti yleistetyt Reedin ja Solomonin koodit.

Määritelmä 2.3. *Olkoon n mikä tahansa kokonaisluku välillä $1 \leq n \leq p^m$. Valitaan $\gamma = (\gamma_1, ..., \gamma_{n-1}) \in \mathbb{F}_{p^m}$ erisuuria alkioita ja $v = (v_1, ..., v_{n-1}) \in \mathbb{F}_{p^m}$ nollasta eroavia alkioita mutta niiden ei tarvitse välttämättä olla erisuuria. Tällöin*

yleistetty Reedin ja Solomonin koodi $GRS_k(\gamma, v) = \{v_1 f(\gamma_1), ..., v_{n-1} f(\gamma_{n-1}) | f \in \mathcal{P}_k\}$*, missä* $\mathcal{P}_k$ *on kaikkien niiden polynomien joukko, joiden aste on alle* k *ja joiden kertoimet ovat kunnasta* $\mathbb{F}_{p^m}$*. [7, s. 176]*

Tämä muistuttaa hieman Reedin ja Solomonin alkuperäisen koodin muodostamistapaa. Nyt eroina on kertoimen v_i lisäksi, että yleistetyn Reedin ja Solomonin koodin pituus voi olla mikä tahansa väliltä $1 \leq n \leq p^m$ sekä se, että kunnnan $\mathbb{F}_{p^m}$ alkioiden $0, 1, \alpha, \alpha^2, ..., \alpha^{p^m-1}$ ei tarvitse olla tässä järjestyksessä vaan ne voivat olla sekaisin. Yleistetyllä Reedin ja Solomonin koodilla on vielä enemmän mahdollisia koodisanoja kuin kapeataajuisella. Tämä kapeataajuinen Reedin ja Solomonin koodi on GRS_k-koodi, jolle $v_i = 1$ kaikilla $1 \leq i \leq p^m - 1$ ja $\gamma_i = \alpha^i$, kun α on kunnan $\mathbb{F}_{p^m}$ primitiivinen alkio.

Yleistetyn GRS_k-koodin virittäjämatriisi on:

$$G_{GRS_k} = \begin{pmatrix} v_1\gamma_1^0 & \dots & v_1\gamma_1^{k-1} \\ v_1\gamma_2^0 & \dots & v_2\gamma_2^{k-1} \\ \vdots & \ddots & \\ v_n\gamma_n^0 & \dots & v_n\gamma_n^{k-1} \end{pmatrix}$$

Alkioita $\gamma_1, ...\gamma_{p^m-1}$ kutsutaan koodin virittäjämatriisin G sarakevirittäjäksi ja alkioita $v_1, ..., v_{p^m-1}$ kutsutaan sarakkeiden kertoimiksi.

2.4 Reedin ja Solomonin koodit ja virhekasaumat

Reedin ja Solomonin koodit ovat tehokkaita virhekasaumien korjaamisessa. Tämä johtuu siitä, että koska koodin aakkostona on kunta $\mathbb{F}_{p^m}$. Kunnan $\mathbb{F}_{p^m}$ alkiot voidaan esittää m-pituisena jonona kunnan $\mathbb{F}_p$ alkioita. Virhekasaumat ovat nimensä mukaisesti peräkkäin tapahtuvia virheitä. Virhekasauma voi esimerkiksi syntyä jos lähetyskanavassa tapahtuu häiriö. Tällöin näiden häiriöhetkien aikana kanavan läpi kulkenut viesti voi vaurioitua ja tapahtuneet virheet ovat peräkkäin.

Esimerkki 2.2. $RS(255, 233)$-koodin avulla koodattu data joka lähetetään kyseisen kanavan läpi, jossa tapahtuu 2 sekunnin häiriö. Tämän koodin alkiot ovat kunnasta $\mathbb{F}_{256}$. Oletetaan, että näiden 2 sekunnin aikana tapahtuu 50 symbolivirhettä. Tiedetään, että $RS(255, 233)$-koodi pystyy korjaamaan 16 virhettä koodisanassa eli koodin virheenkorjaamiskyky ei riitä syntyneiden virheiden

korjaamiseen. Nyt kuitenkin, koska nämä 50 virhettä tapahtuvat peräkkäin ja $RS(255, 233) = RS(2^8 - 1, 233)$-koodin koodisanojen yksittäiset symbolit muodostuvat 8 bitin jonoista, niin nämä 50 virhettä vaikuttavat enintään 7 koodisanan symboliin. Nyt siis $RS(255, 233)$-koodi pystyy korjaamaan häiriön aiheuttamat virheet ja lähetetty data tulee oikein perille.

Usein Reedin-Solomonin koodien aakkostona käytetään jotakin kuntaa $\mathbb{F}_{2^m}$, missä $m \geq 1$, koska nykymaailmassa data on usein binäärisenä. Suosittu RS-koodin pituus on $n = 255$ ja yksi suosittu tämän pituinen koodi on $RS(255, 223)$ eli $RS(2^8-1, 223)$. Kyseisen koodin virheenkorjaamiskyky on $255-223 = 36 = 2t$ eli $t = 16$.

3 Viestin selvittäminen lähetetystä koodisanasta

Kuinka alkuperäinen viesti saadaan selville jonkin kanavan läpi tulleesta koodisanasta, jos matkalla on tapahtunut virheitä? Tähän on olemassa erilaisia tapoja. Käydään ensimmäiseksi läpi alkuperäisellä Reedin ja Solomonin tavalla koodatun koodin purkaminen ja virheiden korjaaminen. Tässä pohjana on käytetty Bhargavan ja Wickerin kirjaa Reed-Solomon Codes and Their Applications [2]. Alkuperäiseen Reedin ja Solomonin koodaustapaan liittyy myös Sudanin ja Guruswamin algoritmi, jonka avulla voidaan korjata enemmän virheitä kuin mitä teoreettinen virheenkorjauskyky olisi.

Syklisille koodeille on olemassa useita algoritmejä, joilla koodeja voidaan purkaa. Yksi näistä tavoista tunnetaan Petersonin, Gorensteinin, Zierlerin koodinpurkamisena, joka on BCH-koodeille suunniteltu tapa ja toimii siis syklisille koodeille. Petersonin, Gorensteinin, Zierlerin tapa käydään tässä läpi kuten Huffman ja Pless esittävät sen kirjassaan Fundamentals of Error-correcting Codes [7].

3.1 Viestin selvittäminen alkuperäisellä Reedin ja Solomonin koodilla koodatusta viestistä

Kuinka alkuperäinen viesti saadaan selville vioittuneesta koodisanasta, kun koodaukseen on käytetty alkuperäistä Reedin ja Solomonin koodaamistapaa? Käydään tämä tapaus tässä yhteydessä läpi.

Ensinnäkin kaikista koodisanoista, jotka ovat muotoa

$$c = (c_0, c_1, ..., c_{p^m-1}) = \{P(0), P(\alpha), ..., P(\alpha^{p^m-1})\}$$

voidaan muodostaa p^m lineaarisen yhtälön ryhmä:

$$
\begin{aligned}
&P(0) = m_0 \\
&P(\alpha) = m_0 + m_1\alpha + m_2\alpha^2 + ... + m_{k-1}\alpha^{k-1} \\
&P(\alpha^2) = m_0 + m_1\alpha^2 + ... + m_{k-1}\alpha^{2(k-1)} \\
&\vdots \\
&P(\alpha^{p^m-1}) = m_0 + m_1\alpha^{p^m-1} + ... + m_{k-1}\alpha^{p^m-1}
\end{aligned} \tag{3.1}
$$

Viesti m voidaan selvittää tästä yhtälöryhmästä. Jos koodisanassa ei ole tapahtunut virheitä, niin koska koodin dimensio on k, joten yhtälöryhmästä voidaan valita mitkä tahansa k yhtälöä, joiden avulla viestisymbolit voidaan selvittää. Valitaan selvyyden vuoksi nyt k ensimmäistä yhtälöä. Valituista k yhtälöstä muodostetaan matriisit:

$$
\begin{pmatrix}
1 & 0 & 0 & \cdots & 0 \\
1 & \alpha & \alpha^2 & \cdots & \alpha^{k-1} \\
1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(k-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \alpha^{k-1} & \alpha^{2(k-1)} & \cdots & \alpha^{(k-1)(k-1)}
\end{pmatrix}
\begin{pmatrix} m_0 \\ m_1 \\ \vdots \\ m_{k-1} \end{pmatrix} =
\begin{pmatrix} P(0) \\ P(\alpha) \\ \vdots \\ P(\alpha^{k-1}) \end{pmatrix} \tag{3.2}
$$

Lasketaan sitten vasemman puoleisimalle matriisille determinantti:

$$
\det \begin{pmatrix}
1 & 0 & 0 & \cdots & 0 \\
1 & \alpha & \alpha^2 & \cdots & \alpha^{k-1} \\
1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(k-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \alpha^{k-1} & \alpha^{2(k-1)} & \cdots & \alpha^{(k-1)(k-1)}
\end{pmatrix} = 1 \cdot \det \begin{pmatrix}
\alpha & \alpha^2 & \cdots & \alpha^{k-1} \\
\alpha^2 & \alpha^4 & \cdots & \alpha^{2(k-1)} \\
\vdots & \vdots & \ddots & \vdots \\
\alpha^{k-1} & \alpha^{2(k-1)} & \cdots & \alpha^{(k-1)(k-1)}
\end{pmatrix}
$$

Huomataan, että oikean puoleinen determinantti saadaan muotoon:

$$
\alpha^{\sum_{j=1}^{k-1} j} \det \begin{pmatrix}
1 & 1 & 1 & \cdots & 1 \\
\alpha & \alpha^2 & \alpha^3 & \cdots & \alpha^{(k-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\alpha^{k-2} & \alpha^{(k-1)} & \alpha^{2(k-1)} & \cdots & \alpha^{(k-2)(k-1)}
\end{pmatrix}
$$

Kysessä on taas Vandermonden matriisi, joka on epäsingulaarinen. Tällöin myös alkuperäinen matriisi on epäsingulaarinen, joten sillä on olemassa käänteismatriisi. Näin ollen on mahdollista laskea arvot $(m_0, ..., m_{k-1})$.

Mistä tahansa k yhtälön joukosta voidaan muodostaa Vandermonden matriisi, joten yhtälöistä 2.1 voidaan todella valita mitkä tahansa viestin m selvittämistä varten.

Nyt jos viestin lähetyksessä tapahtuu t virhettä, niin tällöin t kappaletta yhtälöistä:

$$\begin{aligned}
&P(0) = m_0 \\
&P(\alpha) = m_0 + m_1\alpha + m_2\alpha^2 + ... + m_{k-1}\alpha^{k-1} \\
&P(\alpha^2) = m_0 + m_1\alpha^2 + ... + m_{k-1}\alpha^{2(k-1)} \\
&\vdots \\
&P(\alpha^{p^m-1}) = m_0 + m_1\alpha^{p^m-1} + ... + m_{k-1}\alpha^{p^m-1}
\end{aligned}$$

ovat muuttuneet. Yhtälöistä voidaan valita k kappaletta yhtälöitä $\binom{p^m}{k}$ tavalla. Näistä tavoista kuitenkin $\binom{t+k-1}{k}$ tavalla voidaan valita siten, että joukkoon kuuluu virheellinen yhtälö. Jos kaikille mahdollisille yhtälöryhmille lasketaan ratkaisut ja ratkaisun, jonka todennäköisyys on suurin, oletetaan olevan oikea, niin täytyy päteä $\binom{t+k-1}{k} < \binom{p^m-t}{k}$. Toisin sanoen niiden tapojen määrä, joilla voidaan valita k oikein olevaa yhtälöä on suurempi kuin niiden tapojen, joilla k valitun yhtälön joukossa on ainakin yksi virheellinen yhtälö. Kun binominkertoimet lasketaan auki, niin saadaan, että $t < \frac{p^m-k+1}{2}$. Toisin sanoen Reedin ja Solomonin koodi voi korjata t virhettä kun $t = \left\lfloor \frac{p^m-k+1}{2} \right\rfloor = \left\lfloor \frac{n-k+1}{2} \right\rfloor$.

Edellä kerrotusta seuraa, että jos lähetetystä viestistä halutaan selvittää alkuperäinen viesti, niin silloin täytyy laskea $\binom{p^m}{k}$ eri tavalla yhtälöt 2.1. Tämä voi olla hidasta, koska tapojen määrä kasvaa nopeasti kun parametri n kasvaa.

Esimerkki 3.1. Halutaan lähettää viesti $m = (1, 2, 2, 0, 2, 1, 1, 1)$. Valitaan aakkostokunnaksi $\mathbb{F}_{3^2}$, jonka muodostamisessa käytetään polynomia x^2+x+2, jolloin viestiksi saadaan $m = (\alpha^6, \alpha^5, \alpha^2, \alpha^7) = (m_0, m_1, m_2, m_3)$. Käytettävä koodi on näin ollen $RS(4, 9)$-koodi, joka voi korjata $t = 2$ virhettä. Viestistä saadaan koodisana saadaan kun lasketaan $m(x) = \alpha^6 + \alpha^5 x + \alpha^2 x^2 + \alpha^7 x^3$ arvo jokaiselle aakkostokunnan $\mathbb{F}_9$ alkiolle:

$$
\begin{aligned}
m(0) &= \alpha^6 & \qquad m(\alpha^5) &= \alpha^7 \\
m(\alpha) &= \alpha^7 & m(\alpha^6) &= \alpha^5 \\
m(\alpha^2) &= \alpha^3 & m(\alpha^7) &= \alpha^7 \\
m(\alpha^3) &= \alpha^6 & m(\alpha^8) &= 1 \\
m(\alpha^4) &= \alpha^4
\end{aligned}
$$

Näin koodisanaksi saadaan

$$c = (\alpha^6, \alpha^7, \alpha^3, \alpha^6, \alpha^4, \alpha^7, \alpha^5, \alpha^7, \alpha^0) = (1, 2, 1, 1, 2, 2, 1, 2, 0, 2, 1, 1, 2, 0, 1, 1, 0, 1).$$

Jos nyt tämä koodisana lähetettäisiin meluisan kanavan läpi ja vastaanottajan saama viesti on

$$y = (\alpha^2, \alpha^7, \alpha^3, \alpha^6, \alpha^5, \alpha^7, \alpha^5, \alpha^7, \alpha^0),$$

niin lähetyksessä on tapahtunut kaksi virhettä kohdissa c_0 ja c_4. Nyt täytyisi laskea $\binom{9}{4} = 126$ tavalla yhtälöt (3.1), jotta saataisin alkuperäinen viesti m selville. Tätä ei luonnollisesti tässä tutkielmassa esitetä. Lasketaan kuitenkin viesti $m = (m_0, m_1, m_2, m_3)$ kun valitaan viestistä y neljä ensimmäistä symbolia ja sitten, kun valitaan ensimmäiset neljä oikein olevaa symbolia.

Neljä ensimmäistä symbolia ovat $(\alpha^2, \alpha^7, \alpha^3, \alpha^6)$, jolloin saadaan matriisi

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & \alpha & \alpha^2 & \alpha^3 \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 \\ 1 & \alpha^3 & \alpha^6 & \alpha \end{pmatrix}$$

jonka käänteismatriisi on

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ \alpha^5 & \alpha^7 & \alpha^7 & \alpha^2 \\ \alpha^7 & \alpha^7 & \alpha^3 & \alpha^3 \\ \alpha^6 & \alpha^2 & \alpha^3 & \alpha^7 \end{pmatrix}$$

Tästä saadaan, että $m = (\alpha^2, \alpha^6, \alpha^7, \alpha)$, joka on täysin väärin.

Lasketaan sitten vastaavat laskut kun käytetään oikein olevia arvoja eli valitaan viestistä y symbolit $(y_1, y_2, y_3, y_5) = (\alpha^7, \alpha^3, \alpha^6, \alpha^7)$. Tällöin saadaan matriisi

$$\begin{pmatrix} 1 & \alpha & \alpha^2 & \alpha^3 \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 \\ 1 & \alpha^3 & \alpha^6 & \alpha \\ 1 & \alpha^5 & \alpha^2 & \alpha^7 \end{pmatrix}$$

jonka käänteismatriisiksi saadaan

$$\begin{pmatrix} \alpha^4 & \alpha^2 & \alpha^2 & \alpha^5 \\ \alpha^2 & \alpha^3 & \alpha^4 & \alpha^2 \\ 0 & \alpha^4 & \alpha^4 & \alpha^4 \\ \alpha^6 & \alpha^5 & \alpha^6 & \alpha^3 \end{pmatrix}$$

Tästä saadaan, että $m = (\alpha^6, \alpha^5, \alpha^2, \alpha^7)$, joka on juuri se mikä pitää.

Edellisen esimerkin laskut on laskettu Maximalla. Nämä Maxima-laskut löytyvät Liitteistä 1-3.

3.2 Petersonin, Gorensteinin ja Zierlerin koodinpurkaminen

Olkoon y kanavan läpi tullut viesti. Tiedetään, että $y = c + e$, missä $e = (e_0, ..., e_{n-1})$ on virhevektori. Vektorin alkiot $e_i \in \mathbb{F}_{p^m}$ ilmaisevat tapahtuneen virheen suuruuden. Jos virhettä ei ole tapahtunut, niin $e = 0$ ja $y = c$.

Jotta mahdolliset virheet pystytään korjaamaan, niin oletetaan, että virheellisten symbolien määrälle w pätee $w \leq t$. Tällöin virhevektori on polynomina $e(x) = e_{k_1}x^{k_1} + ... + e_{k_w}x^{k_w}$, missä $k_j, j = 1, .., w$ ovat virheiden koordinaatit. Näitä koordinaatteja ei tunneta vielä.

Alkuperäisen viestin selvittäminen tehdään kolmessa osassa, jotka ovat virhesyndrooman laskeminen, virheenpaikannuspolynomin löytäminen ja sen juurien laskeminen.

3.2.1 Virhesyndrooman laskeminen

Muistetaan, että jos vektori $c \in C$, niin $c(\alpha^i) = 0$ kaikilla $i \in T$. Viestille y pätee, että $y(\alpha^i) = c(\alpha^i) + e(\alpha^i) = e(\alpha^i)$. Jos saatu viesti y on koodisana,

niin $y(\alpha^i) = e(\alpha^i) = 0$ ja jos taas $y(\alpha^i) = e(\alpha^i) \neq 0$, niin tiedetään, että lähetyksessä on tapahtunut virhe. Termiä $y(\alpha^i)$ kutsutaan syndroomaksi ja merkitään $S_i = y(\alpha^i)$. Syndrooma siis riippuu ainoastaan virhevektorista $e(\alpha^i)$.

Ensimmäinen vaihe kohti alkuperäisen koodisanan selvittämistä on laskea vektorille y syndroomat:

$$S_i = y(\alpha^i) = \sum_{j=1}^{w} e_{k_j}(\alpha^i)^{k_j} = \sum_{j=1}^{w} e_{k_j}(\alpha^{k_j})^i, \text{ missä } 1 \leq i \leq 2t. \tag{3.3}$$

Merkitään selvyyden vuoksi, että $e_{k_j} = E_j$, jolloin E_j vastaa virheen suuruutta virhevektorin kohdassa k_j. Olkoon myös $\alpha^{k_j} = X_j$, joka vastaa virheen sijaintia vektorissa e.

$$S_i = \sum_{j=1}^{w} E_j X_j^i, \text{ kun } 1 \leq i \leq 2t. \tag{3.4}$$

Yhtälöistä $(S_1, ..., S_{2t})$ saadaan yhtälöryhmä. Muodostuva yhtälöryhmä on kuitenkin epälineearinen virheiden sijaintien X_j, $j = 1, ..., w$ suhteen. Lisäksi kertoimet E_j ovat tuntemattomat. Yhtälöryhmästä tehdään lineearinen E_j suhteen, jolloin virheiden suuruudet pystytään laskemaan. Tätä varten tarvitaan uusia muuttujia $\sigma_1, ..., \sigma_w$. Seuraavassa osiossa selvitään näiden avulla virheiden sijainnit.

3.2.2 Virheenpaikannuspolynomin selvittäminen

Halutaan selvittää virheiden sijainnit vektorissa y. Määritellään virheen paikannuspolynomiksi

$$\sigma(x) = (1 - xX_1)(1 - xX_2) \cdots (1 - xX_w) = 1 + \sum_{i=1}^{w} \sigma_i x^i. \tag{3.5}$$

Huomataan, että kyseisen polynomin juuret ovat virheiden paikkojen X_j käänteisluvut X_j^{-1} eli $\sigma(X_j^{-1}) = 1 + \sigma_1(X_j)^{-1} + \sigma_2(X_j)^{-2} + \cdots + \sigma_w(X_j)^{-w} = 0$ kaikilla $j = 1, .., w$. Kun $\sigma(X_j^{-1})$ kerrotaan termillä $E_j X_j^{i+w}$ saadaan mille tahansa i, että

$$\begin{aligned} &E_j X_j^{i+w} + \sigma_1 X_j^{-1} E_j X_j^{i+w} + \cdots + \sigma_w X_j^{-w} E_j X_j^{i+w} \\ &= E_j X_j^{i+w} + \sigma_1 E_j X_j^{i+w-1} + \cdots + \sigma_w E_j X_j^i \\ &= 0. \end{aligned}$$

Kun näistä lasketaan summa j:n suhteen saadaan, että

$$\sum_{j=1}^{w} E_j X_j^{i+w} + \sum_{j=1}^{w} \sigma_1 E_j X_j^{i+w-1} + \cdots + \sum_{j=1}^{w} \sigma_w E_j X_j^{i} = 0. \tag{3.6}$$

Huomataan, että nämä summat vastaavat syndroomeja $(S_1, ..., S_{2t})$, missä siis $S_i = \sum_{j=1}^{w} E_j X_j^i$ kun $1 \leq i$ ja $i + w \leq 2t$. Koska luvun alun oletuksen mukaan $w \leq t$, niin saadaan edelleen, että

$$\begin{aligned} &S_{i+w} + \sigma_1 S_{i+w-1} + \sigma_2 S_{i+w-2} + \cdots + \sigma_w S_i = 0 \leftrightarrow \\ &\sigma_1 S_{i+w-1} + \sigma_2 S_{i+w-2} + \cdots + \sigma_w S_i = -S_{i+w} \end{aligned}$$

missä $1 \leq i \leq t$.

Tästä saadaan matriisi

$$\begin{pmatrix} S_1 & \dots & S_w \\ S_2 & \dots & S_{w+1} \\ \vdots & \ddots & \vdots \\ S_w & \dots & S_{2w-1} \end{pmatrix} \begin{pmatrix} \sigma_w \\ \sigma_{w-1} \\ \ddots \\ \sigma_1 \end{pmatrix} = \begin{pmatrix} -S_{1+w} \\ -S_{2+w} \\ \ddots \\ -S_{2w} \end{pmatrix}$$

Nyt tästä matriisiyhtälöstä pitäisi selvittää muuttujat σ_i kaikille $1 \leq i \leq w$. Toisin sanoen pitäisi saada laskettua käänteismatriisi vasemman puoleisimmalle matriisille. Käänteismatriisin olemassaolon varmistamiseen ja siten laskemiseen tarvitaan seuraavaa lausetta.

Lause 3.1. *Olkoon $\mu \leq t$ ja olkoon*

$$M_\mu = \begin{pmatrix} S_1 & S_2 & \cdots & S_\mu \\ S_2 & S_3 & \cdots & S_{\mu+1} \\ \vdots & \ddots & & \\ S_\mu & S_{\mu+1} & \cdots & S_{2\mu-1} \end{pmatrix}$$

Matriisi M_μ on epäsingulaarinen, jos $\mu = w$ ja singulaarinen, jos $\mu > w$.

Todistus. Ei todisteta tässä tutkielmassa. Ks.[7, s. 182] □

Sattuneiden virheiden määrää w ei tiedetä. Tässä tapauksessa se pitää yrittää arvata. Olkoon tämä arvaus nyt μ. Jos arvaus osuu oikeaan, niin matriisi M_μ on epäsingulaarinen eli $\det M_\mu \neq 0$. Arvaaminen kannattaa aloittaa suurimmasta mahdollisesta virheiden määrästä. Olkoon siis $\mu = t$. Jos matriisi M_μ on epäsingulaarinen tällä arvolla voidaan jatkaa seuraavaan vaiheeseen ja ratkaista $\sigma(x)$ juuret. Jos taas matriisi on singulaarinen niin arvataan uudelleen, tällä kertaa $\mu = t - 1$. Edetään näin kunnes löydetään sopiva μ. Siirrytään siis kolmanteen vaiheeseen ja ratkaistaan virheiden paikat.

3.2.3 Virheen paikan ja suuruuden selvittäminen

Edellä määriteltiin, että polynomin $\sigma(x)$ juuret ovat virheiden sijaninti numeroiden käänteisluvut. Juurien löytämiseksi joudutaan usein käymään läpi mekaanisesti kaikki $\sigma(\alpha^i)$ arvot, missä $0 \leq i < n$. Kun kaikki juuret ovat selvillä, niin selvitetään niiden käänteisalkiot X_j. Tämän jälkeen neljäs vaihe alkuperäisen koodisanan selvittämiseksi on laskea yhtälöistä (3.4) virheiden suuruudet E_j. Tällöin saadaan selville polynomi $e(x)$ ja alkuperäinen koodisana saadaan yhtälöstä $c(x) = y(x) - e(x)$.

Esimerkki 3.2. Olkoon $y = (\alpha^0, \alpha^4, \alpha, \alpha^6, \alpha^4, \alpha^0, 0, \alpha^7)$ $RS(8,3)$-koodin avulla lähetetty viesti, joka on polynomina $y(x) = 1 + \alpha^4 x + \alpha x^2 + \alpha^6 x^3 + \alpha^4 x^4 + x^5 + \alpha^7 x^7$.

Nyt syndromeiksi saadaan:

$$\begin{aligned} S_1 &= y(\alpha) = \alpha^5 \\ S_2 &= y(\alpha^2) = \alpha^6 \\ S_3 &= y(\alpha^3) = \alpha^6 \\ S_4 &= y(\alpha^4) = \alpha^5 \end{aligned}$$

Arvataan nyt, että lähetyksen aikana virheitä olisi tapahtunut kahdessa kohdassa, jolloin saadaan matriisi

$$M_2 = \begin{pmatrix} S_1 & S_2 \\ S_2 & S_3 \end{pmatrix} = \begin{pmatrix} \alpha^5 & \alpha^6 \\ \alpha^6 & \alpha^6 \end{pmatrix}$$

Tästä saadaan det $M_2 = \alpha^5 \neq 0$ eli matriisi on epäsingulaarinen. Arvaus oli siis oikea ja virheitä on tapahtunut kahdessa kohtaa. Lasketaan nyt käänteismatriisi M_2^{-1}, jonka avulla voidaan laskea σ_1 ja σ_2 yhtälöstä

$$\begin{pmatrix} \alpha^5 & \alpha^6 \\ \alpha^6 & \alpha^6 \end{pmatrix} \begin{pmatrix} \sigma_2 \\ \sigma_1 \end{pmatrix} = \begin{pmatrix} \alpha^2 \\ \alpha \end{pmatrix}$$

Käänteismatriisiksi saadaan

$$M_2^{-1} = \begin{pmatrix} \alpha & \alpha^5 \\ \alpha^5 & 1 \end{pmatrix}$$

Edelleen saadaan, että

$$\begin{pmatrix} \sigma_2 \\ \sigma_1 \end{pmatrix} = \begin{pmatrix} 1 \\ \alpha^2 \end{pmatrix}$$

Nyt saadaan, että $\sigma(x) = 1 + \sigma_1 x + \sigma_2 x^2 = 1 + \alpha^2 x + x^2$. Kun lasketaan nollakohdat, niin saadaan $x = \alpha^3$ ja $x = \alpha^5$. Koska pätee, että $\sigma(X_i^{-1}) = 0$, niin virheiden paikoiksi saadaan $X_1 = (\alpha^5)^{-1} = \alpha^3$ ja $X_2 = (\alpha^3)^{-1} = \alpha^5$. Nyt voidaan laskea virheiden suuruudet syndromeista S_1 ja S_2:

$$S_1 = E_1 X_1 + E_2 X_2 \leftrightarrow \alpha^5 = \alpha^3 E_1 + \alpha^5 E_2$$
$$S_2 = E_1 X_1^2 + E_2 X_2^2 \leftrightarrow \alpha^6 = \alpha^6 E_1 + \alpha^2 E_2$$

Saadaan $E_1 = \alpha^3$ ja $E_2 = \alpha^2$, jolloin $e(x) = \alpha^3 x^3 + \alpha^2 x^5$. Edelleen saadaan

$$\begin{aligned} c(x) = y(x) - e(x) &= 1 + \alpha^4 x + \alpha x^2 + \alpha^6 x^3 + \alpha^4 x^4 + x^5 + \alpha^7 x^7 - (\alpha^3 x^3 + \alpha^2 x^5) \\ &= 1 + \alpha^4 x + \alpha x^2 + \alpha^5 x^3 + \alpha^4 x^4 + \alpha x^5 + \alpha^7 x^7) \end{aligned}$$

Eli $c = (\alpha^0, \alpha^4, \alpha, \alpha^5, \alpha^4, \alpha, 0, \alpha^7)$. Tämä on sama koodisana kuin esimerkissä 2.1 tavalla 1. viestille $m = (1, 2, 2, 0, 2, 1, 1, 1)$ laskettu koodisana.

Maxima-laskut tähän esimerkkiin löytyvät Liitteesä 4.

Tämä koodaustapa toimii hyvin kun koodin virheenkorjaamiskyky on melko pieni. Jos virheenkorjauskyky kasvaa, niin toinen vaihe käy raskaaksi laskea. Tietokoneellakin hyvin suuren virheenkorjaamiskyvyn tapauksessa laskeminen on aikaavievää kun käsitellään isoja matriiseja. Berlekamp ja Massey sekä Sugiyama, Kasahava, Hirasawa ja Namekawa ovat kehittäneet algoritmejä, joilla tästä Petersonin, Goresteinin ja Zierlerin algoritmin toisesta vaiheesta saadaan selvitettyä virheenpaikannuspolynomi $\sigma(x)$ tehokkaammin. [7, s. 186-195]

3.3 Sudanin ja Guruswamin algoritmi

McElicen artikkelissa The Guruswami-Sudan Decoding Algorithm for Reed-Solomon Codes (2003) on esitelty Guruswamin ja Sudanin algoritmi. Sudan ja Guruswami julkaisivat kyseisen algoritmin Reedin ja Solomonin koodien purkamiseen vuonna 1999. Tässä algoritmissa oletetaan, että koodi on koodattu alkuperäisellä Reedin ja Solomonin koodilla. Kyseinen algoritmi on kiinnostava, koska sen avulla voidaan korjata enemmän virheitä kuin Reedin ja Solomonin koodeille perinteinen $t = \left\lfloor \frac{n-k+1}{2} \right\rfloor$. Kuten aiemmin tutkielmassa ollaan todettu, niin kun ylitetään arvo t niin on mahdollista, että kaksi RS-koodin koodisanaa on yhtä etäällä korjattavasta viestistä y. Arvoa t korkeampi virheenkorjaamiskyky perustuu siihen, että tutkitaan *todennäköisyyksiä* tapahtumalle, että tietylle etäisyydelle $t_e > t$ korjattavasta viestistä y osuu kaksi $RS(n, k)$ koodin koodisanaan. Tämä todennäköisyys voi olla hyvinkin pieni.

Sudanin ja Guruswamin algoritmillä korjattavien virheiden maksimi määrä on $t_{GS} \leq n - 1 - \left\lfloor \sqrt{(k-1)n} \right\rfloor$. Monessa tilanteessa t_{GS} voi olla voi olla huomattavasti suurempi kuin t. Algoritmin avulla voidaan laskea heuristinen yläraja todennäköisyydelle, että t_m-säteelle viestistä y osuu jokin muu koodisana kuin lähetetty c. Näin ollen algoritmin avulla voidaan osoittaa monesti teoreettista suurempi virheenkorjaamiskyky monille Reedin ja Solomonin koodeille.

Algoritmin idea on, että saadusta viestistä y muodostetaan tietyin ehdoin kahden muuttujan polynomi. Tälle polynomille etsitään tekijät, jotka ovat muotoa $y - p(x)$, missä $p(x)$ vastaa koodin C etäisyydellä t_m viestistä y olevaa koodisanaa. Luku m on kokonaisluku siten, että $m \geq 1$. Se on interpolointikerroin, jonka suuruus määräytyy myös tietyin ehdoin. Kahden muuttujan polynomille etsitään tekijöitä luvun m eri suuruksilla ja joka kerralla näistä tekijöistä muodostetaan lista ja näille listoille lasketaan todennäköisyys, että t_m-säteelle viestistä y osuu jokin muu koodisana kuin lähetetty c.

Vaikka algoritmin avulla voidaan osoittaa, että koodin virheenkorjaamiskyky voi olla suurempi kuin perinteinen arvo t, niin oikean koodisanan löytäminen voi olla haastavaa. Sudanin ja Guruswanin julkaisema algoritmiä ei nimittäin oltu suunniteltu käytäntöön vaan enemmänkin juuri osoittamaan, että algoritmi suurempaa virheenkorjaamiskykyä varten on olemassa. Myöhemmin Roth-Ruckenstein ja Kötter ovat parannelleet algoritmia, jonka myötä sitä voidaan käyttää käytännössä.

3.4 Puuttuvan symbolin selvittäminen

Entä jos lähetyksessä yksi tai useampi symboli häviää? Symbolien häviäminen on yleinen virhe. Reedin ja Solomonin koodit pystyvät korjaamaan myös häviämävirheitä. Tässä kappaleessa esitetään lause, joka näyttää Reedin ja Solomonin koodien virheenkorjaamiskyvyn, kun lähetyksen aikana tapahtuu sekä häviämiä että symbolien muutoksia. Varsinaista algoritmia häviämien korjaamiseen ei tässä tutkielmassa näytetä mutta kiinnostunut lukija voi lukea siitä esimerkiksi McEliecen kirjasta Theory of Information and Coding [12]

Häviämä näkyy vektorissa y symbolina $*$. Tätä varten saapuvan viestin akkostoa pitää laajentaa siten, että symboli $*$ on mahdollinen. Tästä saadaan aakkostoksi $\overline{F} = \mathbb{F}_{p^m} \cup \{*\}$. Tämän aakkoston avulla voidaan antaa määritelmä *laajennetulle Hammingin etäisyydelle* [12].

Määritelmä 3.1. *Laajennettu Hammingin etäisyys jonkin joukon F symboleille (x, y)*

$$\overline{d}_H(x,y) = \begin{cases} 0, jos\, x = y \\ 1, jos\, x \neq y\, ja\ x\ tai\ y\ kumpikaan\ ei\ ole\ * \\ \frac{1}{2}, jos\, x \neq y\, ja\ joko\ x\ tai\ y\ on\ * \end{cases}$$

Tämä laajennetaan vektoreille siten, että kun $x = (x_1, ..., x_n)$ ja $y = (y_1, ..., y_n)$, niin

$$\overline{d}_H(x,y) = \sum_{i=1}^{n} \overline{d}_H(x_i, y_i).$$

Nyt kun koodi käytttää virheen korjaamiseen lähimmän naapurin koodausta ja etäisyytenä käytetään laajennettya Hammingin etäisyyttä niin virheenkorjaamiskyvylle saadaan seuraava lause:

Lause 3.2. *Olkoon C $RS(n,k)$ koodi kunnan $\mathbb{F}_{p^m}$ suhteen. Koodi C pystyy korjaamaan e_0 häviämää ja e_1 virhettä jos $e_0 + 2e_1 \leq n - k$.*

Todistus. Muistetaan, että *RS*-koodille $n - k = d_{min} - 1$. Olkoon koodisana c lähetetty meluisan kanavan läpi, jolloin on tapahtunut e_0 häviämää ja e_1 virhettä siten, että $e_0 + 2e_1 \leq n - k = d_{min} - 1$. Olkoon nyt y läpi tullut vektori. Tällöin $\overline{d}_H(c, y) = \frac{1}{2}e_0 + e_1 \leq \frac{1}{2}(d_{min} - 1)$. Nyt, jotta koodi pystyy korjaamaan virheet , niin koodisanan c täytyy olla nyt vektoria y lähinnä oleva koodisana. Näin on, sillä jos jokin toinen koodisana c' olisi lähempänä tai yhtä lähellä, niin pätisi $\overline{d}_H(c', y) \leq \frac{1}{2}(d_{min} - 1)$. Tällöin $\overline{d}_H(c, c') \leq \overline{d}_H(c, y) + \overline{d}_H(y, c') \leq \frac{1}{2}(d_{min} - 1) + \frac{1}{2}(d_{min} - 1) = d_{min} - 1$. Tämä on ristiriita ja nähdään tosiaan, että c on vektorin y lähin koodisana ja y on mahdollista korjata oikein koodisanaksi c. □

Kirjallisuutta

[1] Berlekamp,E.R. 1968. Algebraic Coding Theory. McGraw-Hill Book Company.

[2] Bhargava, V.K, Wicker, S.B. 1994. Reed-Solomon Codes and Their Applications. Wiley-IEEE Press.

[3] Bhattacarya, P.B., Jain, S.K., Nagpaul, S.R. 1994. Basic Abstract Algebra. 2. painos. Cambridge University Press.

[4] Blahut, R.E. 2003. Algebraic Codes for Data Transmission. New York, USA: Cambridge University Press.

[5] Caruso, F., D'Aurizio, J., McAndrew, A., van Nek, V. 2012. Finite Field Computations in Maxima. Internert document. http://cs.swan.ac.uk/∼ csoliver/ok-sat-library/internet_html/doc/doc/Maxima/5.29.0/gf_manual.pdf

[6] Geisel, W.A. 1990. Tutorial on Reed-Solomon error correction coding. NASA Johnson Space Center; Houston, TX, United States.

[7] Huffman, W. C., Pless, V. 2003. Fundamentals of Error-correcting Codes. Cambridge, U.K.: Cambridge University Press.

[8] Jiang, Y. 2010. Practical Guide to Error-Control Coding Using MATLAB. Norwood, USA: Artech House.

[9] Kløve, T. 2007. Codes For Error Detection. River Edge, USA: World Scientific.

[10] Ling, S., Xing, C. 2004. Coding Theory : A First Course. Cambridge, GB: Cambridge University Press.

[11] McEliece, R.J., 2003, The Guruswami-Sudan Decoding Algorithm for Reed-Solomon Codes, IPN PR 42-153.

[12] McEliece, R.J, 1985. The Theory of Information and Coding. 2. laitos. Cambridge, U.K: Cambridge University Press.

[13] MacWilliams, F.J., Sloane, N.J.A. 1977. The Theory of Error-Correcting Codes. 9. laitos. Amsterdam, Netherlands: North-Holland Mathematical Library.

[14] Neubauer, A., Freudenberger, J., Kuhn, V. 2007. Coding Theory : Algorithms, Architectures, and Applications. Chichester, U.K: Wiley-Interscience.

[15] Seroussi, G., Roth, R.M. N.d. On MDS Extensions of Generalized Reed-Solomon Codes. IEEE transactions on information theoyry, vol. IT-32, NO. 3, MAY 1986.

[16] Rocha, V. 2014. Elements of Algebraic Coding Systems. New York, USA: Momentum Press.

[17] vanLint, J.H. 1998. Introduction to Coding Theory. 3.laitos. Berlin, Springer-Verlag.

[18] Xing, Chaoping, and Ling, San. 2004. Coding Theory : A First Course. West Nyack, USA: Cambridge University Press.

LIITTEET

LIITE 1

Esimerkin 3.1 Maxima laskut koodisanan c selvittämiseksi:

```
(%i1)  gf_set_data(3, x^2+x+2);
```

(%o1) $Structure[GF - DATA]$

Asetetaan polynomi m(z)

```
(%i2)  m(z):=gf_primitive()^6+gf_primitive()^5*z
           +gf_primitive()^2*z^2+gf_primitive()^7*z^3;
```

(%o2) $\mathrm{m}\left(z\right) := \mathrm{gf\_primitive}\left(\right)^{6} + \mathrm{gf\_primitive}\left(\right)^{5} z + \mathrm{gf\_primitive}\left(\right)^{2} z^{2} + \mathrm{gf\_primitive}\left(\right)^{7} z^{3}$

Lasketaan polynomin m(z) arvot jokaisella kunnan GF(3^2) alkiolla

```
(%i3)  m(0);
```

(%o3) x^{6}

```
(%i4)  a:makelist(m(gf_primitive()^i), i,1,gf_order());
```

(%o4) $[x^{10} + 2\,x^{6} + x^{4}, x^{13} + x^{7} + 2\,x^{6}, x^{16} + 2\,x^{8} + x^{6}, x^{19} + x^{10} + x^{9} + x^{6}, x^{22} + x^{12} + x^{10} + x^{6}, x^{25} + x^{14} + x^{11} + x^{6}, x^{28} + x^{16} + x^{12} + x^{6}, x^{31} + x^{18} + x^{13} + x^{6}]$

```
(%i5)  b:map(gf_eval,a);
```

(%o5) $[x + 1, 2\,x + 2, x + 2, 2, x + 1, 2\,x, x + 1, 1]$

```
(%i6)  map(gf_log,b);
```

(%o6) $[7, 3, 6, 4, 7, 5, 7, 0]$

Muutetaan alkioiden polynomimuodot pätkämuotoon

```
(%i7)  map(gf_p2l, b);
```

(%o7) $[[1,1],[2,2],[1,2],[2],[1,1],[2,0],[1,1],[1]]$

LIITE 2

Esimerkin 3.1 Maxima laskut kun käytetään viestin y neljää ensimmäistä symbolia:

```
(%i1)  gf_set_data(3, x^2+x+2);
```

(%o1) $Structure[GF - DATA]$

```
(%i2)  gf9s:makelist( gf_eval( (gf_primitive())^j), j,1,gf_order() );
```

(%o2) $[x, 2\,x+1, 2\,x+2, 2, 2\,x, x+2, x+1, 1]$

Muodostetaan sitten matriisi primitiivisen alkion potensseina ilmaistuna

```
(%i3)  m1:matrix([1,0,0,0],
           [1,gf_primitive(),gf_primitive()^2,gf_primitive()^3],
           [1,gf_primitive()^2,gf_primitive()^4,gf_primitive()^6],
           [1,gf_primitive()^3,gf_primitive()^6,gf_primitive()]);
```

(%o3) $$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & x & x^2 & x^3 \\ 1 & x^2 & x^4 & x^6 \\ 1 & x^3 & x^6 & x \end{pmatrix}$$

Muokataan matriisia siten, että se ilmaistaan alkioiden polynomimuotoina

```
(%i4)  m12:matrixmap(gf_eval,m1);
```

(%o4) $$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & x & 2\,x+1 & 2\,x+2 \\ 1 & 2\,x+1 & 2 & x+2 \\ 1 & 2\,x+2 & x+2 & x \end{pmatrix}$$

Lasketaan determinantti ja sen käänteisluku

```
(%i5)  d:determinant(m12);
```

(%o5) $x\left(2\,x-(x+2)^2\right)-(2\,x+1)\,(x\,(2\,x+1)-(x+2)\,(2\,x+2))$
$+(2\,x+2)\,((x+2)\,(2\,x+1)-2\,(2\,x+2))$

```
(%i6)  d:gf_eval(d);
```

(%o6) $2\,x+2$

```
(%i7)  id1:gf_eval(gf_inv(d));
```

(%o7) $2\,x$

Lasketaan adjungoitu matriisi matriisille m1 ja muokataan se taas alkioiden polynomimuotoon

```
(%i8)  adjm1:adjoint(m12);
```

adjm1 matriisimuoto on liian iso tässä näytettäväksi. Seuraavassa kohdassa sievennetään adjm1 matriisi.

```
(%i9)  b:matrixmap(gf_eval,adjm1);
```

(%o9)
$$\begin{pmatrix} 2\,x+2 & 0 & 0 & 0 \\ 1 & 2\,x+1 & 2\,x+1 & 2\,x \\ 2\,x+1 & 2\,x+1 & x+2 & x+2 \\ x & 2\,x & x+2 & 2\,x+1 \end{pmatrix}$$

Lasketaan kaanteismatriisi adjungoidun matriisin ja det(m1) käänteisalkion avulla

```
(%i10) kaanteism11:b*id1;
```

(%o10)
$$\begin{pmatrix} 2\,x\,(2\,x+2) & 0 & 0 & 0 \\ 2\,x & 2\,x\,(2\,x+1) & 2\,x\,(2\,x+1) & 4\,x^2 \\ 2\,x\,(2\,x+1) & 2\,x\,(2\,x+1) & 2\,x\,(x+2) & 2\,x\,(x+2) \\ 2\,x^2 & 4\,x^2 & 2\,x\,(x+2) & 2\,x\,(2\,x+1) \end{pmatrix}$$

```
(%i11) kaanteism1: matrixmap(gf_eval,kaanteism11);
```

(%o11)
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 2\,x & x+1 & x+1 & 2\,x+1 \\ x+1 & x+1 & 2\,x+2 & 2\,x+2 \\ x+2 & 2\,x+1 & 2\,x+2 & x+1 \end{pmatrix}$$

Tarkistetaan, että saatu matriisi on tosiaan m1 käänteismatriisi

```
(%i12) gf_matmult(kaanteism1,m1);
```

(%o12)
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Lasketaan (m_0,m_1,m_2,m_3)

```
(%i13) P1:matrix([2*x+1],[x+1],[2*x+2],[x+2]);
```

$$(\%o13)\quad \begin{pmatrix} 2\,x+1 \\ x+1 \\ 2\,x+2 \\ x+2 \end{pmatrix}$$

```
(%i14) gf_matmult(kaanteism1,P1);
```

$$(\%o14)\quad \begin{pmatrix} 2\,x+1 \\ x+2 \\ x+1 \\ x \end{pmatrix}$$

Saadaan, että m=(a^2,a^6,a^7,a), kun a on primitiivinen alkio

LIITE 3

Esimerkin 3.1 Maxima laskut kun käytetään viestin y neljää ensimmäistä oikein olevaa symbolia:

```
(%i1)  gf_set_data(3, x^2+x+2);
```

(%o1) $Structure[GF - DATA]$

Ilmaistaan matriisi m2 primiriivisen alkion avulla

```
(%i2)  m2:matrix([1,gf_primitive(),gf_primitive()^2,gf_primitive()^3],
                 [1,gf_primitive()^2,gf_primitive()^4,gf_primitive()^6],
               [1,gf_primitive()^3,gf_primitive()^6,gf_primitive()],
               [1,gf_primitive()^5,gf_primitive()^2,gf_primitive()^7]);
```

(%o2)
$$\begin{pmatrix} 1 & x & x^2 & x^3 \\ 1 & x^2 & x^4 & x^6 \\ 1 & x^3 & x^6 & x \\ 1 & x^5 & x^2 & x^7 \end{pmatrix}$$

Muokataan m2 siten, että se ilmaistaan alkioiden polynomimuotojen avulla

```
(%i3)  m2:matrixmap(gf_eval,m2);
```

(%o3)
$$\begin{pmatrix} 1 & x & 2\,x+1 & 2\,x+2 \\ 1 & 2\,x+1 & 2 & x+2 \\ 1 & 2\,x+2 & x+2 & x \\ 1 & 2\,x & 2\,x+1 & x+1 \end{pmatrix}$$

Lasketaan m2 determinantti ja sen käänteisalkio

```
(%i4)  d2:determinant(m2);
```

(%o4) $(2\,x+1)\left(-2\,x^2+(x+1)\,(2\,x+2)-2\,(x+2)-2\,x-1\right)$
$-\,2\left((x+1)\,(2\,x+2)-2\,x^2\right)$
$-\,(2\,x+2)\,((2\,x+1)\,(2\,x+2)-(x-1)\,(2\,x+1)-2\,x\,(x+2)-4)$
$+\,(x+2)\,((2\,x+1)\,(2\,x+2)-2\,x\,(x+2))$
$-\,x\,(-x\,(2\,x+1)+(x+1)\,(x+2)+(x-1)\,(x+2)-2)$
$+\,(2\,x+1)\,((x+1)\,(x+2)-x\,(2\,x+1))$

```
(%i5)  d2:gf_eval(d2);
```

(%o5) $2\,x+1$

```
(%i6)  id2:gf_inv(d2);
```

(%o6) $x+2$

Lasketaan matriisin m2 adjungoitu matriisi

```
(%i7)  adjm2:adjoint(m2);
```

adjm2 matriisimuoto on liian iso, joten sitä ei tässä kohden näytetä. Seuraavassa kohdassa sievennetään matriisi adjm2.

```
(%i8)  b2:matrixmap(gf_eval,adjm2);
```

(%o8) $$\begin{pmatrix} x+2 & 2 & 2 & x+1 \\ 2 & 2\,x & x+2 & 2 \\ 0 & x+2 & x+2 & x+2 \\ 1 & x+1 & 1 & 2\,x \end{pmatrix}$$

Lasketaan käänteismatriisi

```
(%i9)  kaanteism2:b2*id2;
```

(%o9) $$\begin{pmatrix} (x+2)^2 & 2\,(x+2) & 2\,(x+2) & (x+1)\,(x+2) \\ 2\,(x+2) & 2\,x\,(x+2) & (x+2)^2 & 2\,(x+2) \\ 0 & (x+2)^2 & (x+2)^2 & (x+2)^2 \\ x+2 & (x+1)\,(x+2) & x+2 & 2\,x\,(x+2) \end{pmatrix}$$

```
(%i10) kaanteism2:matrixmap(gf_eval,kaanteism2);
```

(%o10) $$\begin{pmatrix} 2 & 2\,x+1 & 2\,x+1 & 2\,x \\ 2\,x+1 & 2\,x+2 & 2 & 2\,x+1 \\ 0 & 2 & 2 & 2 \\ x+2 & 2\,x & x+2 & 2\,x+2 \end{pmatrix}$$

Tarkistetaan, että saatu matriisi todella on m2 käänteismatriisi

```
(%i11) gf_matmult(m2,kaanteism2);
```

(%o11) $$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Lasketaan m=(m_0,m_1,m_2,m_3)

```
(%i12) P2:matrix([x+1],[2*x+2],[x+2],[x+1]);
```

$$(\%o12)\quad \begin{pmatrix} x+1 \\ 2\,x+2 \\ x+2 \\ x+1 \end{pmatrix}$$

```
(%i13) M2:gf_matmult(kaanteism2,P2);
```

$$(\%o13)\quad \begin{pmatrix} x+2 \\ 2\,x \\ 2\,x+1 \\ x+1 \end{pmatrix}$$

```
(%i14) matrixmap(gf_log,M2);
```

$$(\%o14)\quad \begin{pmatrix} 6 \\ 5 \\ 2 \\ 7 \end{pmatrix}$$

Saadaan, että m=(a^6,a^5,a^2,a^7), kun a on primitiivinen alkio

LIITE 4

Esimerkin 3.2 Maxima-laskut:

```
(%i1)  gf_set_data(3,x^2+x+2);
```

(%o1) $Structure[GF-DATA]$

```
(%i2)  c(z):=1+gf_primitive()^4*z+gf_primitive()*z^2+gf_primitive()^5*z^3+gf_prim
```

(%o2) $\mathrm{c}\,(z) := 1+\mathrm{gf\_primitive}\,()^4\,z+\mathrm{gf\_primitive}\,()\;z^2+\mathrm{gf\_primitive}\,()^5\,z^3+\mathrm{gf\_primitive}\,()^4\,z^4+\mathrm{gf\_primitive}\,()\;z^5+0\,z^6+\mathrm{gf\_primitive}\,()^7\,z^7$

```
(%i3)  makelist(gf_eval(c(gf_primitive()^i)),i,1,gf_order());
```

(%o3) $[0,0,0,0,x+2,x,2\,x,2\,x]$

Lasketaan syndroomat:

```
(%i4)  y(z):=1+gf_primitive()^4*z+gf_primitive()*z^2+gf_primitive()^6*z^3+gf_prim
```

(%o4) $\mathrm{y}\,(z) := 1+\mathrm{gf\_primitive}\,()^4\,z+\mathrm{gf\_primitive}\,()\;z^2+\mathrm{gf\_primitive}\,()^6\,z^3+\mathrm{gf\_primitive}\,()^4\,z^4+z^5+\mathrm{gf\_primitive}\,()^7\,z^7$

```
(%i5)  makelist(gf_log(gf_eval(y(gf_primitive()^i))),i,1,gf_order());
```

(%o5) $[5,6,6,5,3,0,7,false]$

Muodostetaan matriisi M_2:

```
(%i6)  M:matrix([gf_primitive()^5,gf_primitive()^6],[gf_primitive()^6,gf_primitiv
```

(%o6) $\begin{pmatrix} x^5 & x^6 \\ x^6 & x^6 \end{pmatrix}$

Lasketaan käänteismatriisi adjongoidun matriisin ja M_2:n determinantin avulla:

```
(%i7)  d:determinant(M);
```

(%o7) $x^{11}-x^{12}$

```
(%i8)  d:gf_eval(d);
```

(%o8) $2\,x$

```
(%i9)  id:gf_inv(gf_primitive()^5);
```

(%o9) $2\,x+2$

```
(%i10) adjM:adjoint(M);
```

(%o10) $$\begin{pmatrix} x^6 & -x^6 \\ -x^6 & x^5 \end{pmatrix}$$

```
(%i11) kaanteisM:gf_primitive()^3*adjM;
```

(%o11) $$\begin{pmatrix} x^9 & -x^9 \\ -x^9 & x^8 \end{pmatrix}$$

```
(%i12) gf_eval(-gf_primitive()^9);
```

(%o12) $2\,x$

```
(%i13) kaanteisM:matrix([gf_primitive(),gf_primitive()^5],[gf_primitive()^5, 1]);
```

(%o13) $$\begin{pmatrix} x & x^5 \\ x^5 & 1 \end{pmatrix}$$

Asetetaan sigma matriisi

```
(%i14) s:matrix([gf_primitive()^2],[gf_primitive()]);
```

(%o14) $$\begin{pmatrix} x^2 \\ x \end{pmatrix}$$

Lasketaan sigma_1 ja sigma_2 arvot

```
(%i15) sigma:gf_matmult(kaanteisM,s);
```

(%o15) $$\begin{pmatrix} 1 \\ 2\,x+1 \end{pmatrix}$$

Muodostetaan sigma(x) ja lasketaan sen nollakohdat

```
(%i16) s(x):=1+gf_primitive()^2*x+x^2;
```

(%o16) $\mathrm{s}\left(x\right):=1+\mathrm{gf\_primitive}\left(\right)^2\,x+x^2$

```
(%i17) makelist((gf_eval(s(gf_primitive()^i))),i,1,gf_order());
```

(%o17) $[x+1,2,0,x+1,0,1,2\,x,2\,x]$

Lasketaan nollakohtien käänteisalkiot

```
(%i18) gf_inv(gf_primitive()^3);
```

(%o18) $2\,x$

```
(%i19) gf_inv(gf_primitive()^5);
```

(%o19) $2\,x + 2$

```
(%i20) gf_log(2*x);
```

(%o20) 5

```
(%i21) gf_log(2*x+2);
```

(%o21) 3