Kaksi pistettä määräävät suoran, kolme paraabelin – miten tämä selittää QR-koodin toiminnan

Jyrki Lahtonen
Turun yliopisto, matematiikan ja tilastotieteen laitos

Tarinani tavoite on selittää, miten yksinkertaisia polynomien ominaisuuksia voidaan soveltaa virheitä korjaaviin koodeihin. Kaksiulotteiset viivakoodit eli ns. QR-koodit [8] lienevät trendikkäin sovellus, jossa nämä ovat käytössä. Samanlaista menetelmää on käytetty myös antamaan CD-levyille niiden kyky toipua naarmuista sekä suojaamaan 1. sukupolven digitelevisiolähetyksiä kanavahäiriöiltä.

Polynomien algebrasta tarvitsemme seuraavaa kahta faktaa.

Lause 1. *Astetta n olevalla polynomilla on enintään n nollakohtaa.*

Lause 2. *Jos $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ ovat xy-tason n pistettä, joille $x_i \neq x_j$ aina, kun $i \neq j$, niin on olemassa sellainen yksikäsitteinen polynomi $f(x)$, jonka aste on $< n$ ja jonka kuvaaja kulkee kaikkien mainittujen pisteiden kautta, eli jolle $f(x_i) = y_i$, $i = 1, 2, \ldots, n$.*

Jälkimmäinen tulos ei ole ehkä ihan ilmeinen, joten perustellaan sitä etenkin erikoistapauksissa. Jos tässä $n = 2$, vaadimme, että $f(x)$ on astetta ≤ 1, jolloin sen kuvaaja on suora, tarkemmin sanottuna pisteiden (x_1, y_1) ja (x_2, y_2) kautta kulkeva suora. Kun $n = 3$, vaadimme, että polynomi on enintään astetta kaksi, jolloin sen kuvaaja on joko suora tai y-akselin suuntaan aukeava paraabeli. Haluttu polynomi löydetään ratkaisemalla kertoimet a, b, c yhtälöryhmästä $y_i = ax_i^2 + bx_i + c_i$, $i = 1, 2, 3$.

Korkeampiasteisissa tapauksissa etsitty polynomi löydetään käyttämällä ns. *Lagrangen interpolaatiokaavaa [6],[7].*

Polynomin yksikäsitteisyys seuraa joka tapauksessa Lauseesta 1. Jos nimittäin $f_1(x)$ ja $f_2(x)$ olisivat kaksi eri polynomia, kumpikin astetta $< n$, ja kummankin kuvaaja kulkisi valittujen n pisteen kautta, niin erotuspolynomi $f_1(x) - f_2(x)$ olisi sekin astetta $< n$ ja kaikki luvut x_i, $i = 1, 2, \ldots, n$, olisivat sen nollakohtia. Tämä on Lauseen 1 mukaan mahdotonta.

Tämänkertaiset polynomien sovellukset käsittelevät lähettäjän ja vastaanottajan välistä kommunikointia. Lähettäjä haluaa lähettää vastaanottajalle jonkin viestin. Oletamme, että he ovat etukäteen sopineet tavasta koodata viestit jonoksi lukuja $y_1, y_2, \ldots, y_k$, missä k on jokin luonnollinen luku. Helppo tapa kommunikoida on tällöin tietenkin se, että lähettäjä yksinkertaisesti kirjoittaa viestin johonkin *tiedonsiirtokanavaan*, jota vastaanottaja voi sitten myöhemmin lukea. Kanava voi olla vaikka luokkahuoneessa kädestä käteen kulkeva paperinpala tai älypuhelimen näytöllä oleva QR-koodi. Se voi myös olla vaikka radiotaajuuskaista tai valokuitu – kanavan fyysinen luonne ei meitä nyt kiinnosta.

Jos kaikki menee hyvin, vastaanottaja saa viestin luettavakseen alkuperäisessä muodossaan. Mutta ehkä jo-

kin merkki vain luetaan väärin, tai ehkä kaverin hikinen peukalo tuhrasi yhden luvuista viestin kulkiessa hänen kauttaan. Mitä tehdä? Perusidea on lisätä viestiin ylimääräisiä *tarkistuslukuja* (usein puhutaan tarkistussymboleista). Pidennämme k-lukuisen viestin luvun n-lukuiseksi, $n > k$, tavalla, joka antaa vastaanottajalle mahdollisuuden toipua muutamista hämminkiä aiheuttavista tilanteista.

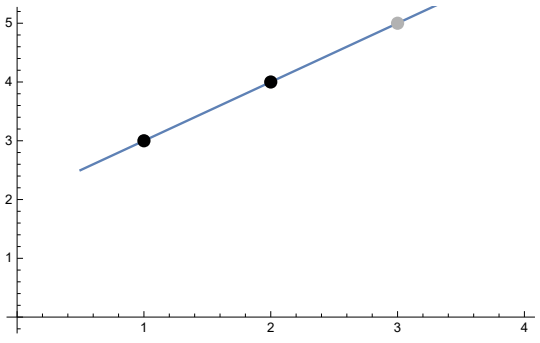
Yksi hyödyllinen tapa toimia on seuraava:

- Sovitaan etukäteen jono (erisuuria) x-koordinaatteja $x_1, \ldots, x_n$.
- Viestiä $y_1, y_2, \ldots, y_k$ ajatellaankin pisteiden

$$P_1 = (x_1, y_1), P_2 = (x_2, y_2), \ldots, P_k = (x_k, y_k)$$

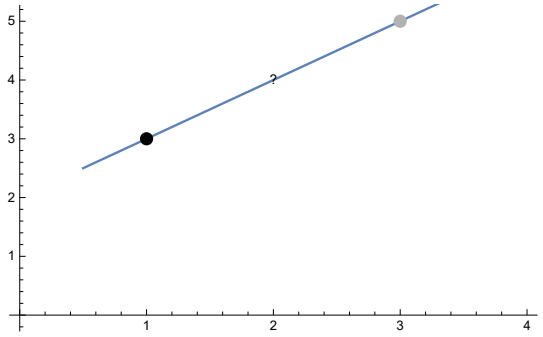
  jonona.
- Määrätään alussa mainittu yksikäsitteinen polynomi $f(x)$ astetta $< k$, jonka kuvaaja kulkee kaikkien pisteiden P_i, $i = 1, 2, \ldots, k$, kautta.
- Laajennetaan viesti sisältämään n lukua $y_1, y_2, \ldots, y_n$ siten, että $y_i = f(x_i)$ kaikille $i = 1, 2, \ldots, n$. Huomaa, että varsinainen viesti (= payload-luvut) muodostuu k:sta ensimmäisestä luvusta, loput $r = n - k$ ovat *tarkistuslukuja*, joita käytetään virhetilanteista toipumiseen.

Alla olevissa kuvissa on yksinkertaisuuden vuoksi aina $x_1 = 1, x_2 = 2, \ldots, x_n = n$. Tarkastellaan ensin leluesimerkkinä tapausta $k = 2$, $n = 3$. Polynomimme ovat siis astetta ≤ 1 ja niiden kuvaajat ovat suoria. Valitaan lähetettäväksi viestiksi $(y_1, y_2) = (3, 4)$. Näemme heti, että tässä tapauksessa polynomi $f(x) = x + 2$, sillä $f(1) = 3 = y_1$ and $f(2) = 4 = y_2$. Koska $f(3) = 5$, laajennetuksi (= koodatuksi) viestiksi saadaan $(y_1, y_2, y_3) = (3, 4, 5)$. Kuvassa 1 payload-lukuja vastaavat pisteet ovat mustia ja tarkistuslukuja vastaavat pisteet harmaita. Hahmottamisen helpottamiseksi mukana on myös polynomin $f(x)$ kuvaaja.



Kuva 1: Kahdesta luvusta muodostuvaan viestiin lisätään kolmas (harmaa) tarkistusluku siten, että muodostuvat kolme pistettä ovat samalla suoralla.

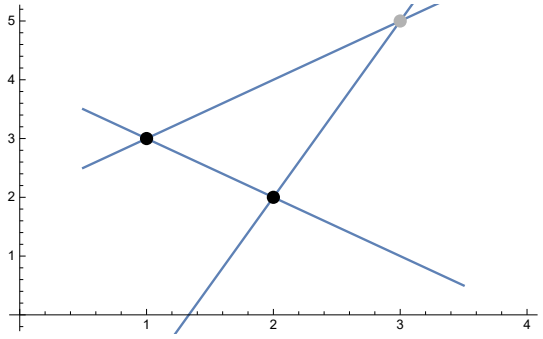
Mitä hyötyä tästä yhdestä tarkistusluvusta meille on? Pohdiskellaanpa ensin tapausta, jossa keskimmäinen luvuista todellakin muuttui kanavassa lukukelvottomaksi. Tällöin vastaanottaja pystyy ainoastaan lukemaan jonon $(3, ?, 5)$. Systeemimme auttaa vastaanottajaa päättelemään, että lukukelvottoman luvun on pakko olla juuri 4. Hän tietää pisteiden $(1, 3)$, $(2, ?)$ ja $(3, 5)$ kaikkien olevan samalla suoralla, ja voi näin ollen määrätä polynomin $f(x)$, ja edelleen tuhoutuneen luvun $y_2 = 4$. Kuva 2 yrittää havainnollistaa tätä.



Kuva 2: Kolmilukuiseksi koodatun viestin keskimmäinen luku on pyyhkiytynyt lukukelvottomaksi, mutta selviää, koska se on samalla suoralla.

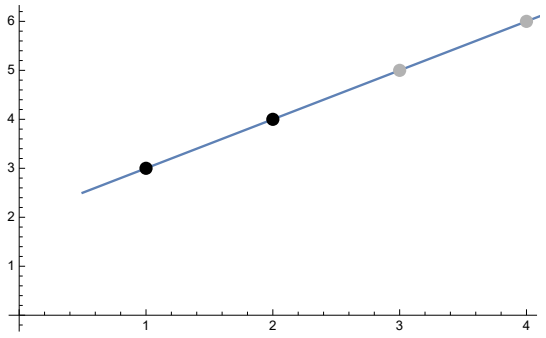
Huomaa, että on täysin merkityksetöntä, mikä kolmesta luvusta oli tuhriutunut. Kunhan vastaanottaja tuntee suoralta kaksi pistettä, hän pystyy sen avulla päättelemään puuttuvan luvun, ja näin lukemaan viestin.

Minimalistinen koodimme ei kuitenkaan suojaa meitä toisenlaisilta virhetilanteilta. Oletetaan seuraavaksi, että vastaanottaja lukee yhden luvuista väärin (ehkä kyseessä oli lukuvirhe tai ehkä kyseinen luku oli kanavassa muuttunut toiseksi). Tällöin hän joutuu luovuttamaan. Vastaanottaja toki huomaa, etteivät hänen lukemansa kolme pistettä ole samalla suoralla. Hän siis tietää jonkin menneen pieleen. Hän ei kuitenkaan pysty päättelemään, mikä luvuista on virheellinen. Minkä tahansa kahden pisteen kautta kulkee suora, eikä hänellä ole mitään keinoa valita kolmen vaihtoehtoisen suoran joukosta. Tilanne kuvassa 3.



Kuva 3: Kun vastaanottajan kolme pistettä eivät ole samalla suoralla, ei voida selvittää, mikä pisteistä on virheellinen.

Miten tällaista virhetilannetta vastaan voidaan suojautua? Osoittautuu, että tarkistuslukujen määrää kasvattamalla selviämme pulmasta! Kokeillaanpa kahden tarkistusluvun käyttöä: $k = 2$, $n = 4$, $r = 4 - 2 = 2$. Payload-viesti saa edelleen olla $(y_1, y_2) = (3, 4)$. Tällä kertaa lisäämme viestiin neljännen pisteen $(x_4, y_4) = (4, f(4)) = (4, 6)$, joten koodattu viesti onkin $(3, 4, 5, 6)$. Koodattu viesti on Kuvan 4 tapainen.



Kuva 4: Kun kahden luvun viestiin lisätään kaksi tarkistuslukua saadaan 4 pistettä samalta suoralta.

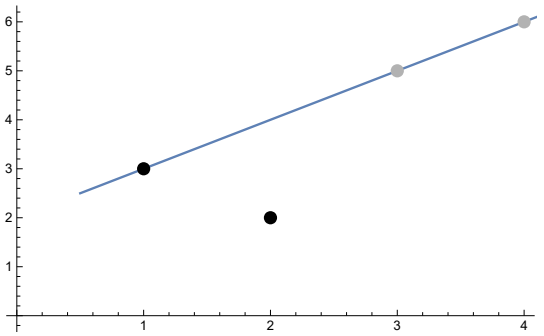
Jos nyt yksi luvuista luetaan väärin, niin vastaanottaja ei ole yhtä avuton. Muut kolme pistettä nimittäin ovat samalla suoralla!

Lause 3. *Jos P_1, P_2, P_3, P_4 ovat tason eri pisteitä, niin enintään yksi suora kulkee ainakin kolmen pisteen kautta.*

Todistus. Tehdään vastaoletus, että voisi olla olemassa kaksi eri suoraa L_1 ja L_2, jotka molemmat sisältävät ainakin kolme valituista pisteistä. Tällöin enintään yksi pisteistä P_1, P_2, P_3, P_4 ei ole suoralla L_1. Sama koskee suoraa L_2, joten suorilla L_1 ja L_2 on vähintään kaksi yhteistä pistettä. Mutta näiden kahden pisteen kautta kulkee vain yksi suora. Ristiriita! □

Jos leluesimerkissämme kanava tai vastaanottaja jälleen tekee yhden virheen, ja lukee $(y_1, y_2, y_3, y_4) = (3, 2, 5, 6)$, niin kuvaa 5 vilkaisemalla on helppo tehtävä tunnistaa, mikä pisteistä on virheellinen. Se olisi helppoa, vaikka en olisikaan lisännyt kuvaan polynomin $y = f(x)$ kuvaajaa.

Yleistetään leluesimerkkiä todenmukaisempiin tilanteisiin. Oletetaan, että olemme täydentäneet k kappaletta payload-lukuja sisältävän viestin kuvatulla tavalla n-lukuiseksi. Mitä ongelmatilanteita vastaan tämä koodi suojaa viestimme? Yllä törmäsimme kahdenlaisiin pulmiin. Jos vastaanottaja ei saa selvää jostakin luvusta, puhutaan *pyyhkiytyneestä* luvusta. Tällöin vastaanottaja kuitenkin tietää vaurioituneen pisteen etukäteen sovitun x-koordinaatin, vain y-koordinaatti puuttuu. Toinen virhetilanne oli *virhe*, väärin luettu luku. Leluesimerkkien perusteella jo arvaamme, että virhe on vakavampi kuin pyyhkiytyminen. Tarvitsimme kaksi tarkistuslukua virheen korjaamiseksi, kun taas pyyhkiytyneestä selvisimme yhdellä.



Kuva 5: Kahden tarkistusluvun avulla voimme korjata yhden virheen, koska löytyy vain yksi suora, joka kulkee kolmen viestiin kuuluvan pisteen kautta.

Oletetaan, että haluamme suojautua t:tä virhettä ja e:tä pyyhkiytymistä vastaan. Montako tarkistuslukua tällöin tarvitsemme? Kaiken kaikkiaan koodatussa viestissä on $n = k + r$ lukua. Pyyhkiytymisten jälkeen on kuvaajan pisteistä vielä tiedossa $n - e$ kappaletta. Näiden pitäisi sitten jotenkin määrätä yksikäsitteinen polynomi, jonka aste $< k$. Vastauksen antaa seuraava Lauseen 3 yleistys.

Lause 4. *Jos tunnemme xy-tasolta $k+2t$ pistettä, joiden x-koordinaatit eroavat, on olemassa enintään yksi astetta $< k$ oleva polynomi $f(x)$, jonka kuvaajalta puuttuu enintään t kappaletta annetuista pisteistä.*

Todistus. Nyt ryhdyn ilkeäksi ja jätänkin tämän harjoitustehtäväksi. Käytä Lausetta 2, kuten käytin faktaa ”kahden pisteen kautta kulkee vain yksi suora” Lauseen 3 todistuksessa. □

Seuraus 4.1. *Kun lisäämme viestiin r kappaletta tarkistuslukuja, muodostuva virheenkorjauskoodi selviää tilanteesta, jossa on e kappaletta pyyhkiytyneitä ja t kappaletta virheellisesti luettuja lukuja aina, kun*

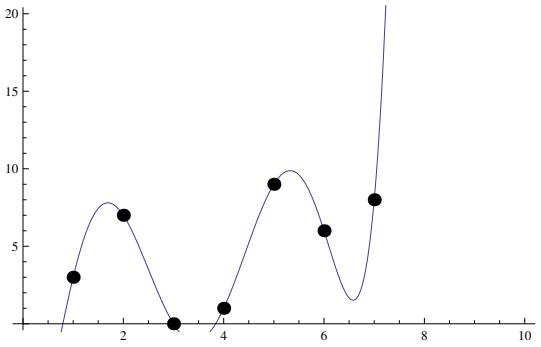
$$r \geq 2t + e.$$

Näemme siis, että tarkistussymboleja lisäämällä voimme suojata viestimme hyvinkin useita virheitä vastaan. On kuitenkin pidettävä mielessä, että tarkistussymbolien lisääminen ei ole ilmaista. Jokainen tarkistusluku nimittäin vaatii yhtä paljon tilaa kanavassa kuin payload-lukukin. Näin ollen virheenkorjauskoodia käyttävän järjestelmän suunnittelijan tulee arvioida yksittäisen pyyhkiytymisen ja/tai virheen todennäköisyys, ja sitten valita koodin parametrit siten, että Seurauksen 4.1 epäyhtälö on voimassa riittävän suurella todennäköisyydellä.

Lukukäsitteen aiheuttamia ongelmia

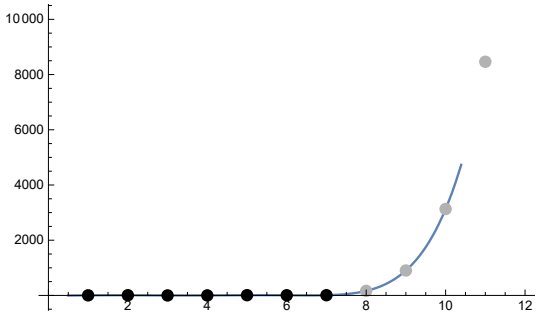
Kuvaamassani skeemassa on muutama vakava ongelma. Yllä puhuin vain *luvuista* spesifioimatta tarkemmin

millaisia lukuja tarkoitan. Kuvien perusteella voi muodostua mielikuva, että luvut voisivat olla mitä tahansa reaalilukuja. Valitettavasti tämä ei ole mahdollista. Perussyy tähän on, että reaaliluvun arvon ilmaiseminen äärettömän tarkasti vaatii äärettömän monen bitin käyttämistä lukua kohti. QR-koodeissa ja CD-levyissä tieto on kuitenkin kirjoitettu *tavuina* eli 8 bitin yksikköinä. Jos harrastat tietokoneohjelmointia, tiedät varmaankin, että tavua voidaan ajatella kokonaislukuna b, joka on välillä $0 \leq b \leq 255 = 2^8 - 1$. Vaikka rajoittaisimmekin payload-luvut välille $[0, 255]$, meillä ei ole takeita siitä, että tarkistusluvut pysyisivät tällä välillä. Valitaanpa esimerkiksi $k = 7$, $r = 4$ ja payload-viestiksi $(y_1, \ldots, y_7) = (3, 7, 0, 1, 9, 6, 8)$. Tämä viesti on kuvassa 6.



Kuva 6: 7-lukuista viestiä vastaa kuudennen asteen polynomin kuvaaja.

Missä ovat harmaat tarkistuspisteet? Kuvaajasta ehkä arvaatkin, että polynomi $f(x)$ kasvaa voimakkaasti viimeisen payload-luvun jälkeen. Määräämällä polynomin $f(x)$ voimme laskea tarkistusluvut $f(8) = 164$, $f(9) = 903$, $f(10) = 3129$ ja $f(11) = 8464$. Meillä on siis vakava ylivuoto-ongelma (overflow). Alivuoto (underflow) on sekin mahdollinen, sillä tarkistusluvuista voi hyvinkin tulla negatiivisia. Varmuuden vuoksi sama kuva 7 eri mittakaavassa.



Kuva 7: Sama kuvaaja niin, että tarkistusluvutkin näkyvät.

Tämä on vakava ongelma siksi, että emme halua käyttää tarkistusluvun kirjaamiseksi yhtään sen enempää bittejä kuin payload-luvunkaan. Tämän ongelman ratkaisemiseksi tarvitaan hiukan yliopistotason algebraa. Lukuteorian alkeita tuntevat voivat tässä vaiheessa tuntea kiusausta käsitellä 'tavua' kokonaislukujen jäännösluokkana modulo 256, ks. esim. [1], [3], eli jäännösluokkarenkaan $R = \mathbb{Z}_{256}$ alkiona, jolloin polynomien arvot lasketaankin käyttäen jakojäännösten algebraa eli modulaarista aritmetiikkaa [2]. Valitettavasti algebralliset oletuksemme eivät tällöin ole voimassa. Jo Lause 1 menee dramaattisesti pieleen. Ensimmäisen asteen polynomilla $f(x) = 16x$ on nimittäin peräti 16 nollakohtaa renkaassa R, sen arvot $f(0), f(16), f(32), \ldots$ antavat kaikki jakojäännökseksi nollan luvulla 256 jaettaessa. Algebraa opiskelemalla huomaat, että tässä sylttytehtaana on se, ettei R ole niin sanottu *kunta*, eli siellä ei ole määritelty kaikkia peruslaskutoimituksia, erityisesti jakolasku ei aina ole mahdollista. Esimerkiksi Turun yliopiston 1. ja 2. vuoden algebran kursseilla opit, että on kuitenkin olemassa kunta $\mathbb{F}_{256}$, jossa on tarkalleen 256 alkiota. Ratkaisu näihin kaikkiin ongelmiin on siis, että 'luku' tarkoittaakin kunnan $\mathbb{F}_{256}$ alkiota. Jätämme tällaisten *äärellisten kuntien* esittelyn toiseen kertaan. Leluesimerkkinä voit tutustua neljän alkion kuntaan vanhassa Solmu-artikkelissa [4] tai Wikipediassa [5].

Lopuksi

Koodausteoria on matematiikan ja tietoliikennetekniikan rajamaastossa oleva tutkimusala. Yksi sen perusongelmista on suunnitella kuvatun kaltaisia menetelmiä toipua kanavahäiriöiden aiheuttamista virheistä. Kuvailemani menetelmä tunnetaan Reedin–Solomonin koodina (tai lyhyesti RS-koodina) keksijöidensä Irving S. Reedin [9] ja Gustave Solomonin [10] kunniaksi. Heidän tapansa valita käytetyt x-koordinaatit mahdollistaa (harmaiden) tarkistuslukujen tehokkaan laskemisen. Voidaan mm. käyttää yksinkertaista polynomien kertolaskua laskennallisesti hankalamman interpolaatiopolynomin asemesta, tai käyttää tehtävään erityisesti soveltuvaa virtapiiriä (ns. lineaarinen siirtorekisteri). Esimerkiksi RS-koodeja käsittelevä Wikipedia-artikkeli [11] esittelee koodien algebran tavalla, joka paremmin soveltuu tarkistuslukujen laskemiseen. Lopputulos on kuitenkin sama.

Käytännössä on tärkeää, että koodausteoria myös tarjoaa RS-koodatun viestin vastaanottajalle tehokkaan tavan paikantaa ja korjata virheet sekä täyttää pyyhkäistyt luvut. Neljän pisteen leluesimerkissämme tämä oli helppoa – riitti etsiä sellainen pistepari, jonka kautta kulkevalla suoralla oli myös jokin kolmas piste. Koska pistepareja oli vain $\binom{4}{2} = 6$, tämä oli nopeaa. Itse asiassa kaikkia pistepareja ei tarvitse edes käydä läpi. Mutta jos käytämmekin viisi virhettä korjaavaa RS-koodia, jossa on $k = 240$ payload-lukua ja $r = 10$ tarkistuslukua, niin vastaavalla alkeellisella logiikalla toimiva vastaanottaja joutuisi tarkistamaan $\binom{250}{10} = 219005316087032475$ interpolaatiopolynomia,

kukin astetta 239 yhtä lähetettyä viestiä kohti. Onneksi koodausteoreetikkojen työkalupakista löytyy paljon tehokkaampia menetelmiä.

A happy thought: algebran työkalut tuovat järjestystä tietoliikenteen kaaokseen.

Viitteet

[1] T. Metsänkylä: ”Kongruenssi – lukuteorian kätevä apuväline”, Solmu 1/1998.

[2] *Kongruenssi*, `https://fi.wikipedia.org/wiki/Modulaarinen_aritmetiikka`

[3] V. Latvala, P. Smolander: ”Modulaarisista laskutaulukoista”, Solmu 2/2003.

[4] K. Ranto, P. Rosendahl: ”Neljän alkion kunta, solitaire-peli ja taikaneliöt”, Solmu 2/2005.

[5] *äärellinen kunta*, `https://fi.wikipedia.org/wiki/äärellinen_kunta`

[6] H. Apiola: ”Polynomit, interpolaatio ja funktion approksimointi”, Solmu 3/2004.

[7] *Lagrangen Interpolaatiopolynomi*, `https://fi.wikipedia.org/wiki/Lagrangen_interpolaatiopolynomi`

[8] *QR-koodi*, `https://fi.wikipedia.org/wiki/QR-koodi`

[9] Irving Reed, `https://en.wikipedia.org/wiki/Irving_S._Reed`

[10] Gustave Solomon, `https://en.wikipedia.org/wiki/Gustave_Solomon`

[11] *RS-koodit*, `https://en.wikipedia.org/wiki/Reed-Solomon_error_correction`