Gode rank-1 latticereglar av høg dimensjon

Masteroppgåve i Anvendt og Rekneorientert matematikk

Vegard Topphol

Matematisk institutt
Universitetet i Bergen

1. juni 2011

Føreord

Denne masteroppgåva vart gjennomført og skriven ved matematisk institutt, avdeling for anvendt og rekneorientert matematikk 2010 til 2011. Den omhandlar numeriske integrasjonsreglar for integrering over mange dimensjonar, og særskild latticereglar av gitt trigonometrisk grad. Ein god del av arbeidet her er dedikert særskild til utvikling og testing av algoritmer for å finne gode rank 1 latticereglar av utvida grad $\delta = 5$ for høg dimensjon.

Fyrst vil eg takke vegleiar Tor Sørevik for å føreslå temaet for masteroppgåva, som det har vore både interessant og utfordrande å arbeide med. Takk også for samtalar, råd om oppgåva og svar på spørsmål når eg stod fast, og for at du stilte alle dei spørsmåla som gjorde at eg sjølv måtte tenkje godt gjennom detaljane i arbeidet.

Vidare vil eg takke alle medstudentane på matematisk institutt for avbrekk frå arbeidet og selskap i kaffepausane. Takk for eit godt og sosialt arbeidsmiljø og for at de viser at det framleis finnast liv der ute.

Til slutt vil eg takke mine kjære foreldre Kjersti og Arne Kåre Topphol for alltid å vere der for meg og for å vise interesse i oppgåva eg har skrive. Takk for at akkurat *de* er mine foreldre.

Innhald

Figurar

Tabellar

1 Innleiing

1.1 Integrering i mange dimensjonar

Innanfor fleire ulike fagfelt, mellom anna statistisk mekanikk, atomfysikk, finansmatematikk, og andre stader, har ein bruk for å kunne integrere over mange dimensjonar. For eit lågt tal på dimensjonar er dette relativt greit og krev ikkje for stor reknekraft. For enkle integral kan ein til og med gjere det analytisk utan alt for store problem.

Men det store fleirtal av funksjonar er dessverre ikkje slik. Dei kan ha for mange dimensjonar, og derfor vere for uoversiktlige til å kunne integrerast analytisk. I tillegg kan dei ha singularitetar eller vere diskontinuerlege, dei kan ha hurtige oscillasjonar, eller generellt sett vere for komplekse til å kunne forståast analytisk. Ofte har ein også med integrandar å gjere som er anten heilt eller delvis ukjend. Integrasjonsområdet kan også vere av ein slik art, med omsyn både på form og avgrensing, at ein ikkje har nokon enkel måte å integrere dei på. I desse tilfella vert det kravd meir spesialiserte integrasjonsmetodar.

Det finnast allereie ein heil del metodar for numerisk integrasjon, der dei eindimensjonale integrasjonsreglane naturleg nok er best utforska. Vi har her to integrasjonsreglar som er verdt å nemne her. Den fyrste av desse er Gaussisk kvadratur som integrerer polynomiske funksjonar eksakt opp til gitt orden. Den andre som her peikar seg ut er trapesregelen, som gir god evaluering av integral av periodiske funksjonar. Ein naturleg tanke er å utvide konsepta vi har i eindimensjonale reglar til å også gjelde for fleire dimensjonar. Den enklaste måten å gjere dette på er å nytte produktintegrasjonsreglar. Dette går heilt enkelt ut på å redusere eit s-dimensjonalt integral til eit sett av s eindimensjonale integral. For få dimensjonar går dette heilt greit, og ein får brukbare resultat utan for store krav til utrekningskraft. Men når ein kjem opp i ikkje så alt for mange dimensjonar vert dette snart ein utilstrekkjeleg strategi. Med dimensjonane veks talet på utrekningar eksponensiellt, og allereie for fem dimensjonar vil ein med enkel produktintegrasjon vere heilt på randa av det som er praktisk mogleg.

Vi treng derfor meir sofistikerte integrasjonsmetodar. Dei metodane vi har tilgjengelege deler seg hovudsakleg i to kategoriar, der hovudproblemet i alle tilfelle reduserer til å velje evalueringspunkt på smarte måtar så feilen blir så liten som mogleg, med så få utrekningar som mogleg. Den fyrste av desse kategoriane er Monte Carlo metodane [17], som går ut på å velge evalueringspunkt tilfeldig og ta gjennomsnittet av desse. I realiteten vil det vere eit sett av psevdotilfeldige punkt ein brukar, og integrasjonsmetoden kan aldri vere betre enn dei beste generatorane av psevdotilfeldige punkt. Ein utnyttar at ved tilfeldig utveljing så vil ein, om ein har eit stort nok utval, nærme seg den reelle middelverdien, og dermed den eksakte evalueringa av integralet. Fordelen med dette er at det går relativt fort å få relativt gode resultat. Ulempen er at det er ein tilfeldig prosess, og ein kan derfor aldri garantere for gode resultat.

Eit alternativ til Monte Carlo metodane er den store gruppa av integrasjonsmetodar som passande nok vert kalla kvasi Monte Carlo metodar[19]. I motsetnad til Monte Carlo metodane fungerer desse ved at ein vel sample punkt ut i frå system konstruert spesifikt for å minske integrasjonsfeilen. Desse metodane er ofte tilpassa enkelte grupper

av integrandar, og er derfor mindre generelle enn Monte Carlo metodar. Om ein kjenner integranden kan derfor eit godt val av ein kvasi Monte Carlo metode vere betre enn rein Monte Carlo integrering. I tillegg, i og med at Monte Carlo integrering i realiteten nyttar psevdotilfeldige punkt, vil dei ha ein viss regularitet som ein ikkje alltid har kontroll over effekten av. Ved å nytte kvasi Monte Carlo metodar kan ein seiast å ta kontroll ovevr denne regulariteten og helder utnytte den for høgare presisjon.

I tillegg til desse to grunnleggjande gruppene av metodar kan det vere verdt å nemne adaptive metoder[23]. Dette er meir måtar å implementere numerisk integrering enn sjølvstendige integrasjonsmetodar, der ein søkjer å forbetre resultata til ein av dei grunnleggjande metodane, typisk ved å nytte feilestimat til å finne endå betre tilnærmingar. Eit enkelt eksempel på dette vil vere å ta endå fleire sample i problemområde for å kunne jamne ut feilen her.

I denne oppgåva vil eg konsentrere meg om ein særskild type integrasjonsreglar kalla latticereglar[16][23]. Dette er ein form for kvadraturreglar, eller også kalla kubaturreglar, og dei kan sjåast på som ein generalisering av den eindimensjonale trapesregelen. Latticereglar er særskild eigna for ein-periodiske funksjonar, men med høve til å tilpassast andre, ikkje-periodiske funksjonar. For enklare å handtere definisjonar og konstruksjonar av latticereglar vil integrasjonsområdet for alle latticereglane vi ser på vere den s-dimensjonale einingskuba C^s. Sidan vi kan skalere eit kvart rektangulært område til ei kube vil desse kvadraturreglane også kunne tilpassast slike integrasjonsområder.

Mitt mål med denne oppgåva er å studere eit sett av latticereglar konstruert på ein spesiell måte ved hjelp av golomblinjalar. Vi vil utvikle algoritmer for å effektivt kunne konstruere rank-1 latticereglar av gitt trigonometrisk grad med lågast mogleg tal på evalueringspunkt. Dette vil bli forklart seinare i teksten. Det har vist seg at ein kan forutsjå grada til latticereglar konstruert ved hjelp av golomblinjalar. Desse algoritmene vil eg så nytte for å finne latticereglar av høg grad og høg dimensjon.

2 Introduksjon til latticereglar

Eg vil her presentere definisjonar og bakgrunnsteori om lattice og latticereglar som er nødvendig for denne oppgåva. Ei gjennomgåande skildring av latticereglar kan finnast hos Sloan og Joe [23], og meir kompakt hos Lyness [16]. Teorien bak latticereglar har vore studert lenge, og kan seiast å ha starta med 'metoden for gode lattice punkt' hos Korobov [12] og Hlawka [11], og seinare også hos Conroy [4]. Ein generalisert definisjon av latticereglar dukka fyrst opp hos Frolov [10] og gjenoppdaga og utvikla vidare av Sloan og Kachoyan [24][25] og av Sloan [22]. Min gjennomgong av bakgrunnsteorien vil stort sett fylgje Sloan og Joe [23].

2.1 Lattice

Basisen for einkvar latticeregel, som også har gitt dei namnet, er eit lattice. Det er latticet som definerer korleis ein for ein spesifik latticeregel vil velje integrasjonspunkta. For å forstå latticereglane må vi derfor definere kva eit lattice er. Enkelt forklart er eit lattice eit periodisk grid av punkt i ein eller fleire dimensjonar. Formelt sett kan vi definere det slik.

Definisjon 2.1 (Lattice)**.** Eit s-dimensjonalt lattice Λ er ei diskret mengde $\{\mathbf{x}\} \in \mathbb{R}^s$ som er lukka under addisjon og subtraksjon.

Vi kan umiddelbart ut i frå denne definisjonen finne nokon av eigenskapane til eit lattice. For det fyrste ser vi at punktet $\mathbf{0} = (0, 0, \ldots, 0)$ altid er med i eitkvart lattice. Dette ser vi ut i frå at dersom $\mathbf{x} \in \Lambda$ må vi også ha at $\mathbf{x} - \mathbf{x} \in \Lambda$. I tillegg er eitkvart punkt $\mathbf{y} = k\mathbf{x}$ der $k \in \mathbb{Z}$ også med i latticet. Latticet som består av alle punkt med heiltalskomponentar vert kalla einingslatticet og vi skriv det som Λ_0^s. Eit minimalt sett av lineært uavhengige punkt $\{\mathbf{x}_i\}$, der alle punkt i latticet Λ kan uttrykkjast som $\mathbf{y} = \sum_i c_i \mathbf{x}_i; c_i \in \mathbb{Z}$ vert kalla generatorar for latticet.

Vidare, på grunn av additivitetskravet til latticepunkta, er alle lattice utanom $\mathbf{0}$ uavgrensa.

Til eitkvart lattice kan det definerast eit dualt lattice.

Definisjon 2.2 (Dualt lattice)**.** Eitkvart lattice Λ har eit dualt lattice $\Lambda^\perp$ definert som settet

$$\{\mathbf{h} \in \mathbb{R} : \mathbf{h} \cdot \mathbf{x} \in \mathbb{Z} \forall \mathbf{x} \in \Lambda\}$$

For einingslatticet får vi at det duale latticet blir einingslatticet sjølv.

Ei spesiell gruppe av lattice er kalla integrasjonslattice. Desse lattice er viktige i samanhang med latticereglar, og vi kan definere dei som lattice innehaldande einingslatticet Λ_0.

Definisjon 2.3 (integrasjonslattice)**.** Eit s-dimensjonalt integrasjonslattice Λ er eit lattice definert av ei mengde

$$\{n_i, \mathbf{z}_i : i = 1, \ldots, t\},$$

der t og n_i er positive heiltal og $\mathbf{z}_i \in \mathbb{Z}^s$, og $\Lambda_0^s \in \Lambda$.

Λ inneheld alle punkt på forma

$$\mathbf{p} = \sum_{i=1}^{t} \frac{\nu_i \mathbf{z}_i}{n_i} \tag{1}$$

der ν_i er eit vilkårleg heiltal.

Ein konsekvens av at Λ_0^s er element i integrasjonslatticet er at alle element i det tilsvarande duale latticet er heiltalsvektorar. Sidan einkvar kanonisk einingsvektor $\mathbf{e}_i \in \Lambda_0^s \in \Lambda$ og $\mathbf{h} \in \Lambda^\perp$ får vi at $\mathbf{h} \cdot \mathbf{e}_i = h_i$, som ut i frå definisjonen må vere heiltal. Vidare vil vi også ha at integrasjonslatticet må vere 1-periodisk, og i tillegg vil alle punkt av typen $\mathbf{p} \mod 1$ liggande i einingskuba vere element i latticet.

I resten av oppgåva vil alle lattice Λ vere integrasjonslattice og $\Lambda^\perp$ det tilsvarande duale lattice.

2.2 Latticereglar

Latticereglar er medlemer av den store familien av numeriske integrasjonsmetodar som vert kalla kvasi-Monte Carlo metodar, og den er spesielt eigna for å integrere kontinuerlege, periodiske integrandar. Standard integrasjonsområde er C^s, den s-dimensjonale einingskuba. Integralet vi vil evaluere vert då

$$If = \int_{C^s} f d\mathbf{x}. \tag{2}$$

Ein latticeregel kan definerast på fleire ulike måtar, noko avhengig av korleis ein definerer eit lattice.

Definisjon 2.4 (Latticeregel)**.** Ein latticeregel er ein integrasjonsregel på forma

$$Qf = \frac{1}{N} \sum_{\mathbf{x}_j \in \Lambda \cap [0,1)^s} f(\mathbf{x}_j) \tag{3}$$

der Λ er eit integrasjonslattice.

Alternativt, dersom ein tek utgangspunkt i likning (1), kan ein skrive dette som

$$Qf = \frac{1}{n_1 n_2 \dots n_t} \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} \cdots \sum_{j_t=0}^{n_t-1} f(j_1 \frac{\mathbf{z_1}}{n_1} + j_2 \frac{\mathbf{z_2}}{n_2} + \cdots + j_t \frac{\mathbf{z_t}}{n_t}) \tag{4}$$

der $\mathbf{z_i}$ er s-dimensjonale heiltalsvektorar, og n_i og t er positive heiltal. Dersom alle $n_i > 1$ vert dette kalla ein t-syklus form [26], og latticet vert sagt å vere av rank t. Denne forma er nyttig i praktisk bruk. Legg merke til at om integranden er 1-periodisk, treng vi ikkje å krevje at $\mathbf{x}_j$ ligg innanfor einingskuba.

I dette arbeidet konsentrerer eg meg om reglar av rank 1, og likning (4) reduserer til

$$Qf = \frac{1}{N} \sum_{j=0}^{N-1} f(j \frac{\mathbf{z}}{N}). \tag{5}$$

Sidan (3) også skal gjelde for rank 1 reglar har vi at alle $\mathbf{z}_i$ der $\mathbf{z}_i/N \in \Lambda \cap [0,1)^s$ må kunne skrivast som

$$\mathbf{z}_i = i\mathbf{z} (\text{mod } N). \tag{6}$$

Vi ser ut i frå dette at talet på evalueringspunkt er nært knytt til storleikne på N. For denne oppgåva har vi i tillegg at alle latticereglane kan definerast ut i frå ein $\mathbf{z} = (1, z_2, \ldots, z_s)$, og vi vil derfor ha at for ein gitt N vil alle latticereglar ha nøyaktig N evalueringspunkt.

Umiddelbart vil ein kanskje seie at ved å konsentrere seg om rank 1 latticereglar vil ein utelukke mange gode reglar av høgare rank. Men ved søk har det vist seg at mange av dei beste reglane er nettop rank 1 reglar, og for dei tilfella der ein har høgare rank beste reglar ligg tilsvarande rank 1 reglar ikkje langt etter [27]. Fordelen med rank 1 vert då openbar ved at dei er mindre komplekse og dermed lettare både å finne fram til og implementere. Ved å konsentrere seg om rank 1 kan ein redusere søkjeromet betydeleg.

Sidan eit integrasjonslattice må vere 1-periodisk, vil det duale latticet til ein rank 1 regel vere N-periodisk, og det vil redusere til

$$\{\mathbf{h} \in \mathbb{Z}^s : \mathbf{h} \cdot \mathbf{z} \equiv 0 (\text{mod } N)\}.$$

Dette kan vi sjå ved å la

$$\mathbf{h}' = \mathbf{h} + cN\mathbf{e}_i \in \Lambda^\perp; 1 \leq i \leq s, c \in \mathbb{Z}$$

der $\mathbf{e}_i$ er ein kanonisk einingsvektor. Vi får då at

$$\begin{aligned} \mathbf{h}' \cdot \mathbf{x} &= h_1 x_1 + \cdots + h_i x_i + cN x_i + \ldots h_s x_s \\ &= \mathbf{h} \cdot \mathbf{x} + cN x_i. \end{aligned}$$

På grunn av (1) får vi at $cNx_i \in \mathbb{Z}$ og $\mathbf{h}' \cdot \mathbf{x} \in \mathbb{Z}$, som fører til at $\Lambda^\perp$ er N-periodisk.

2.3 Trigonometrisk grad

Eg har allereie nemnd at latticereglar er spesiellt tilpassa periodiske funksjonar, og det er derfor naturleg å ville framstille integranden som ein sum av trigonometriske monom i s dimensjonar, analogt med Fourierrekkjer i ein dimensjon.

Definisjon 2.5 (Trigonometrisk monom)**.** Eit s-dimensjonalt trigonometrisk monom er ein funksjon på forma

$$f(\mathbf{x}) = e^{2\pi i \mathbf{h} \cdot \mathbf{x}} \tag{7}$$

der $\mathbf{h} \in \mathbb{Z}^s$

Eit trigonometrisk polynom vert då ein lineær kombinasjon av trigonometriske monom. Vi kallar romet av alle trigonometriske polynom i s dimensjonar $\mathbb{T}^s$. Grada til eit trigonometrisk monom vert $d = ||\mathbf{h}||_1$ og til eit trigonometrisk polynom vert den lik grada til det monomleddet med høgast grad. Romet av alle trigonometriske polynom i s dimensjonar og opp til grad d vert kalla $\mathbb{T}_d^s$.

Dette er praktisk fordi vi kan bruke det til å gi eit tal på kor god ein latticeregel er. Trigonometrisk grad er eit slikt mål som nettop baserer seg på at ein kan tilnærme ein funksjon med eit endeleg trigonometrisk polynom.

Definisjon 2.6 (Trigonometrisk grad for integrasjonsreglar)**.** Ein integrasjonsregel har trigonometrisk grad $d(Q)$ dersom $Qf = If$ for alle $f \in \mathbb{T}_d^s$ og det eksisterer minst ein $g \in \mathbb{T}_{d+1}^s$ slik at $Qg \neq Ig$.

I ein del samanhangar har det vist seg nyttig å definere eit nytt mål, kalla utvida trigonometrisk grad. Dette er definert som $\delta = d(Q) + 1$. Reint matematisk har det lite å seie, men ein del relasjonar blir enklare og ryddigare om vi brukar dette målet.

Slektskapen til eit anna mål, kalla algebraisk grad, er openbar.

Definisjon 2.7 (Algebraisk grad for integrasjonsreglar)**.** Ein integrasjonsregel har algebraisk grad $g(Q)$ dersom $Qf = If$ for alle

$$f \in \sum_{j_1+\cdots+j_s \leq g} a_{j_1 \ldots j_s} x_1^{j_s} \ldots x_s^{j_s}$$

Her er ei innsikt frå lågare dimensjonar nyttig. Alle som har forsøkt seg med interpolasjon med algebraiske polynom veit at ein for høgare grads polynom får store problem med artifaktar i randområda. Dette er ein effekt som ikkje vil forsvinne ved høgare dimensjonar og ein kan derfor tenkje seg at trigonometrisk grad i mange tilfelle er betre og meir presist enn algebraisk grad.

2.4 Tilnærmingsfeil for latticereglar

Motivasjonen for å nytte trigonometrisk grad finn vi i den underliggande teorien for Fourieranalyse. Den seier at ein kvar kontinuerleg og periodisk funksjon i s dimensjonar kan uttrykkjast ved hjelp av ein uendeleg sum av trigonometriske monom.

$$f(\mathbf{x}) = \sum_{\mathbf{h} \in \mathbb{Z}^s} \hat{f}(\mathbf{h}) e^{2\pi i \mathbf{h} \cdot \mathbf{x}}. \tag{8}$$

Ved å nytte latticeregelen på denne likninga kan vi finne integrasjonsfeilen for latticeregelen. Resultatet vert då som i teoremet under.

Teorem 2.1 (Tilnærmingsfeil for latticereglar)**.** *La Q vere ein s-dimensjonal latticeregel, og Λ tilhøyrande integrasjonslattice. Anta vidare at funksjonen f har absolutt konvergent Fourierrekkje. Då har vi at*

$$Qf - If = \sum_{\mathbf{h} \in \Lambda^\perp \setminus \{0\}} \hat{f}(\mathbf{h}). \tag{9}$$

Her er $\hat{f}(\mathbf{h})$ fourierkoeffissienten i $\mathbf{h}$ og $\Lambda^\perp$ er det duale latticet til Λ.

Prov For å sjå at dette stemmer brukar vi likning (8) saman med definisjonen av $\Lambda^\perp$. Vi må vise at

$$Q[e^{2\pi i \mathbf{h} \cdot \mathbf{x}}] = \begin{cases} 1 & \text{for } \mathbf{h} \in \Lambda^\perp \\ 0 & \text{elles} \end{cases}$$

For fyrste delen av provet ser vi at når $\mathbf{h} \in \Lambda^\perp$ har vi ut i frå definisjonen at $\mathbf{h} \cdot \mathbf{x} \in \mathbb{Z}$, som gir oss vidare at $Q[e^{2\pi i \mathbf{h}\cdot\mathbf{x}}] = 1$.

For siste delen av dette provet må vi bruke at integranden er 1-periodisk. Vi kan definere ein operator T_j for 1-periodiske funksjonar.

$$T_j f(\mathbf{x}) = f(\mathbf{x} + \mathbf{x}_j); \mathbf{x}_j \in \Lambda \cap [0,1)^s.$$

Vi får for eit gitt lattice eksakt N ulike translasjonar tilsvarande latticepunkta i eiingskuba. Dette gir vidare at

$$T_k T_j f(\mathbf{x}) = f(\mathbf{x} + \mathbf{x}_j + \mathbf{x}_k) = f(\mathbf{x} + \mathbf{x}_l)$$

der $\mathbf{x}_l$ er den unike vektoren i Λ og $[0,1)^s$ som har ein avstand til $\mathbf{x}_j + \mathbf{x}_k$ på ein heiltalsvektor. På grunn av denne unikskapen får vi då som resultat at $\{T_k T_0, \ldots, T_k T_{N-1}\}$ berre gir translasjonane ei ny rekkjefylgje. Gjennomsnittet av alle translasjonane $T_j f$ vert

$$\bar{f} = \frac{1}{N} \sum_{i=0}^{N-1} T_j f$$

Vi kan vidare sjå at $\bar{f}$ er invariant under translasjonane i og med at

$$T_k \bar{f} = \frac{1}{N} \sum_{j=0}^{N-1} T_k T_j f = \frac{1}{N} \sum_{l=0}^{N-1} T_l f = \bar{f}.$$

Vi definerer ein ny funksjon $g_{\mathbf{h}}(\mathbf{x}) = e^{2\pi i \mathbf{h}\cdot\mathbf{x}}$ med $\mathbf{h} \in \mathbb{Z}^s$, $\mathbf{x} \in \mathbb{R}^s$. Ved å nytte translasjonen på denne får vi at

$$T_k g_{\mathbf{h}} = e^{2\pi i \mathbf{h}\cdot\mathbf{x}_k} g_{\mathbf{h}}.$$

Vi ser her at dersom $\mathbf{h} \notin \Lambda^\perp$, så må $g_{\mathbf{h}} \neq T_k g_{\mathbf{h}}$ for ein eller annan $0 \leq k \leq N-1$. I tillegg har vi at

$$\bar{g}_{\mathbf{h}} = \frac{1}{N} \sum_{j=0}^{N-1} T_j g_{\mathbf{h}} = \left(\frac{1}{N} \sum_{j=0} e^{2\pi i \mathbf{h}\cdot\mathbf{x}_j} \right) g_{\mathbf{h}}.$$

Gjennomsnittet er invariant under transformasjonen T_k. Samstundes kan vi finne ein k som gjer at $g_{\mathbf{h}}$ ikkje er invariant under den same transformasjonen. Dette gir oss dermed at uttrykkjet i parentesen må vere lik 0. ■

Ein konsekvens av dette vert då, som tidlegare nemnd, at latticereglar passar særdeles godt for funksjonar med raskt avtakande fourierkoeffisientar. Dette gjeld mellom anna for funksjonar som er 1-periodiske.

Det naturlege neste steget vert då å finne ut korleis ein skal minske denne feilen. Vi veit at når den 1-periodiske utvidinga av ein funksjon f er k gongar deriverbar, så vil fourierkoeffissientane ha ein konvergensrate $\hat{f}(h) \sim 1/h^k$. Dersom utvidinga av

funksjonen er uendeleg deriverbar, noko som kun gjeld for funksjonar som sjølv er 1-periodiske, vil vi få at konvergensen bli $\hat{f}(h) \sim C^{-h}; |C| > 1$. I det s-dimensjonale problemet får vi at dersom f er 1-periodisk og dei partiellderiverte

$$\frac{\partial^{q_1+\dots q_s} f}{\partial x_1^{q_1} \dots \partial x_s^{q_s}}, 0 \leq q_k \leq \alpha, 1 \leq k \leq s$$

eksisterer og er kontinuerlege på $[0,1)^s$ er

$$\hat{f} \sim 1/(\bar{h}_1 \bar{h}_2 \dots \bar{h}_s)^\alpha. \tag{10}$$

der $\bar{h} = \max(1, |h|)$ For uendeleg deriverbare funksjonar i s dimensjonar får vi $\hat{f}(\mathbf{h}) \sim C_1^{-|h_1|} C_2^{-|h_2|} \dots C_s^{-|h_s|}$. Dersom $C_1 = C_2 = \dots = C_s$ får vi

$$\hat{f}(\mathbf{h}) \sim C^{-||\mathbf{h}||_1}. \tag{11}$$

Ut i frå dette kan vi sjå at dei mest signifikante koeffisientane høyrer til $||\mathbf{h}||_1 < d$. På grunn av (10) kan vi skrive feilestimatet som

$$Qf - If = c \sum_{\mathbf{h} \in \Lambda^\perp \setminus \{0\}} \frac{1}{(\bar{h}_1 \bar{h}_2 \dots \bar{h}_s)^\alpha}.$$

For integrasjonsreglar vert konsekvensen av teorem (2.1) og argumentasjonen over, at integralet av (8) må bli lik 0 for alle heiltalsvektorar $\mathbf{h}$ der $0 < ||\mathbf{h}||_1 \leq d$.

$$\begin{cases} Q[1] & = I[1] & = 1 \\ Q[\exp(2\pi i \mathbf{h} \cdot \mathbf{x})] & = I[\exp(2\pi i \mathbf{h} \cdot \mathbf{x})] & = 0 \quad \forall 0 < ||\mathbf{h}||_1 \leq d \end{cases} \tag{12}$$

2.5 Ekvivalens mellom latticereglar

Når vi studerer latticereglar er det nyttig å merke seg visse likskapar mellom beslekta latticereglar. Dette er noko vi kan utnytte for å gjere søk etter latticereglar meir effektive. Særskild nyttig er det å sjå etter likskapar som gjer at latticereglar har lik trigonometrisk grad. For denne oppgåva er det nyttig å definere ekvivalensklasser ut i frå to transformasjonar.

Definisjon 2.8 (Ekvivalensklasse)**.** To vektorar $\mathbf{z}$ og $\mathbf{z}'$ dannar latticereglar i same ekvivalensklasse dersom

1) $\mathbf{z}'$ er danna ved permutasjon av komponentane i $\mathbf{z}$

2) og\eller $\mathbf{z}'$ er danna av $\mathbf{z}$ ved å erstatte z_i med $N - z_i$.

Latticereglar som er beslekta på denne måten har nokre felles eigenskapar. Det viktigaste for denne oppgåva er at dei vil ha same trigonometrisk grad.

Teorem 2.2. *Dersom* $\mathbf{z}$ *og* $\mathbf{z}'$ *høyrer til same ekvivalensklasse, og* $\mathbf{z}$ *dannar ein latticeregel av utvida grad* δ*, så vil også* $\mathbf{z}'$ *danne ein latticeregel av utvida grad* δ

Prov Vi vil fyrst vise at $\mathbf{z}$ og $\mathbf{z}'$ er ekvivalente med omsyn til trigonometrisk grad under permutasjon. Vi definerer ein operator $P_{i,j}(\mathbf{z}) = \mathbf{z}'$ der komponentane z_i og z_j har byt plass. $P_{i,j}$ er matrisen som avvik frå identitetsmatrisen I, ved at komponentane $(i,i) = (j,j) = 0$, medan $(i,j) = (j,i) = 1$, og vi har i tillegg $P_{ij} = P_{ij}^{-1} = P_{ji}$ og $P_{ij}(\mathbf{z})$ er invertibel. Vi antar at det eksisterer ein $\mathbf{h} \in \mathbb{Z}^s$ slik at

$$(\mathbf{h} \cdot \mathbf{z}) \mod N = 0,$$

$$= (h_1 z_1 + \cdots + h_i z_i + \cdots + h_j z_j + \cdots + h_s z_s) \mod N. \tag{13}$$

For $\mathbf{z}'$ får vi då vidare dersom $\mathbf{h}' = P_{i,j}(\mathbf{h})$, at

$$(\mathbf{z}' \cdot \mathbf{h}') \mod N$$

$$= (h_1 z_1 + \cdots + h_j z_j + \cdots + h_i z_i + \cdots + h_s z_s) \mod N = 0.$$

Sidan P er invertibel vil kvar $\mathbf{h}$ ha, for gitte $0 < \{i,j\} \leq s$, ein unik $\mathbf{h}' = P_{i,j}(\mathbf{h})$, og fordi P er ein rein permutasjon av komponentane vil $||\mathbf{h}'||_1 = ||\mathbf{h}||_1$. Dersom $\mathbf{z}$ genererer eit lattice av utvida grad δ har vi ein $\mathbf{h}$ der $||\mathbf{h}||_1 = \delta$ som gir (13). Dette fører til at latticet generert av $\mathbf{z}'$ også har utvida grad δ.

Vi går vidare til å vise at ved å erstatte z_i med $N - z_i$ vil vi også få to latticereglar av lik grad. Vi let $\mathbf{z} = (z_1, \ldots, z_i, \ldots, z_s)$ og $\mathbf{z}' = (z_1, \ldots, N - z_i, \ldots, z_s)$. Vi antar vidare, som over, at vi for ein $\mathbf{h} \in \mathbb{Z}$ har at

$$(\mathbf{h} \cdot \mathbf{z}) \mod N$$

$$= (z_1 h_1, \ldots, z_i h_i, \ldots, z_s h_s) \mod N = 0. \tag{14}$$

Vi kan vidare setje $\mathbf{h}' = (h_1, \ldots, -h_i, \ldots, h_s)$, og på grunn av (14) får vi

$$(\mathbf{h}' \cdot \mathbf{z}') \mod N$$

$$= [z_1 h_1, \ldots, (N - z_i)(-h), \ldots, z_s h_s] \mod N$$

$$= [(z_1 h_1) + \cdots - (N h_i) + (z_i h_i) + \cdots + (z_s h_s)] \mod N = 0.$$

Ved same argumentasjon som over får vi då at dersom $\mathbf{z}$ genererer eit lattice av utvida grad δ vil også $\mathbf{z}'$ generere eit lattice av utvida grad δ. ■

Nokre medlemmer i desse ekvivalensklassene er i ulike samanhangar nyttigare enn andre [28].

Teorem 2.3. *Einkvar ekvivalensklasse av rank 1 latticereglar inneheld ein representant som tilfredsstiller*

$$0 < z_1 < z_2 < \cdots < z_s < N/2,$$

og ein representant som tilfredsstiller

$$N/2 < z_1 < z_2 < \cdots < z_s < N.$$

Prov For ein vilkårleg $\mathbf{z}$ der $\mathbf{z}/N \in \Lambda$ vil $(\mathbf{z}(\text{mod } N))/N \in \Lambda$. Dette gir oss at alle $z_i < N$. Ved å nytte transformasjonane i definisjon 2.8 kan vi vise begge påstandane i teoremet. Vi permuterer komponentane i $\mathbf{z}$ og får $z_1 < z_2 < \cdots < z_s$. Dersom $z_i > N/2$ for ein eller annan $0 < i \leq s$ let vi $z_i' = N - z_i < N/2$. Elles kan vi setje $z_i' = z_i$. Vi har med det ein $\mathbf{z}'$ ekvivalent med $\mathbf{z}$ der $0 < z_1 < z_2 < \cdots < z_s < N/2$.

Sidan alle $0 < z_i' < N/2$ kan vi igjen vidare setje $z_i'' = N - z_i'$ og får ein $\mathbf{z}''$ ekvivalent med $\mathbf{z}'$ og $\mathbf{z}$ der $N/2 < z_1 < z_2 < \cdots < z_s < N$. ■

3 Konstruksjon av gode latticereglar

3.1 Gode latticereglar

Målet med denne oppgåva er som nemnd å søkje etter gode latticereglar. Men for å vite kva som er gode latticereglar må eg presisere kva ein optimal latticeregel er.

Definisjon 3.1 (Optimal latticeregel av grad d og dimensjon s)**.** Ein optimal latticeregel av grad d i s dimensjonar er ein latticeregel som tilfredsstiller $\min N(\Lambda)$ for alle Λ slik at $d(\Lambda) = d$.

Dersom integranden er uendeleg deriverbar vil det vere fornuftig, ut i frå (11), å bruke 1-normen til $\mathbf{h}$ som kvalitetsmål, der ein vil maksimere storleiken

$$\delta(Q) = \min_{\mathbf{h} \in \Lambda^\perp / \{0\}} (||\mathbf{h}||_1)$$

for at $C^{-|\mathbf{h}|_1}$ skal bli så liten som mogleg. I denne oppgåva har vi valt å anta at at integranden er glatt og 1-periodisk, vil det derfor vere naturleg å bruke dette målet, som tilsvarar trigonometrisk grad nemnd tidlegare.

Basert på desse kriteria finnast det fleire ulike angrepsvinklar når ein vil leite etter og konstruere gode latticereglar. Om ein konsentrerer seg om rank 1 reglar har vi i hovudsak tre måtar. Den fyrste går på at ein kan la N vere konstant og så variere $\mathbf{z}$ vektoren for å finne den latticeregelen som av desse har høgast grad. Alternativt kan ein for ein gitt $\mathbf{z}$ variere N for å optimalisere grada. Begge desse framgongsmåtane har sine fordelar. For begge gjeld det at ein må ha ein effektiv måte å rekne ut trigonometrisk grad på. Den fyrste metoden har den fordelen at ein alltid kan finne reglar med eit lågt tal på evalueringspunkt. Ulempa er at dersom ein vil ha høg grad kan det krevje at ein må auke N, og optimaliseringsproblemet vert då eit grensesetjingsproblem. Fordelen med den siste metoden er at når ein fyrst har funne ein god $\mathbf{z}$ kan ein enkelt auke N fram til ein har ynska grad. Problemet her vert då å finne reglar for korleis $\mathbf{z}$ skal sjå ut for å garantere gitt grad. Den tredje måten vil vere å fastsetje grad først, for så å variere både $\mathbf{z}$ og N. Denne framgongsmåten kompenserer til ei viss grad for problema med dei to andre metodane, men han vil i si reinaste form risikere å kunne krevje eit nokså stort søkjeområde. Eg vil i denne oppgåva konsentrere meg om den siste metoden, men for å unngå alt for store søk vil eg konstruere alle $\mathbf{z}$ ut i frå gitte reglar skildra under. Desse konstruksjonane vil også gi naturlege avgrensingar for søkjeområdet til N.

3.2 Utrekning av trigonometrisk grad

Motivert av uttrykket i likningane (12) kan ein finne ein metode for å rekne ut trigonometrisk grad effektivt. Argumentasjonen her er ei forenkling av den gitt av Ronald Cools [5] for latticereglar av vilkårleg rank, og vil her gjelde berre for reglar av rank 1. Den fyrste likninga $Q(1) = I(1) = 1$ er triviellt tilfredsstilt. Kvadraturet reduserer her til å ta

gjennomsnittet av ein konstant funksjon. Den andre må vi skrive om.

$$0 = Q(e^{2\pi i \mathbf{h}\cdot\mathbf{x}}) = \frac{1}{N}\sum_{j=0}^{N-1} \exp(j\frac{2\pi i \mathbf{h}\cdot\mathbf{z}}{N})$$

$$\Updownarrow$$

$$0 = \sum_{j=0}^{N-1} \exp(j\frac{2\pi i \mathbf{h}\cdot\mathbf{z}}{N})$$

Ein latticeregel har då per definisjon utvida grad δ dersom

$$\forall \mathbf{h} : 0 < ||\mathbf{h}||_1 < \delta, 0 = \sum_{j=0}^{N-1} \exp(j\frac{2\pi i \mathbf{h}\cdot\mathbf{z}}{N})$$

som, ut i frå Teorem 2.1 og definisjonen 2.2 av det duale latticet Λ^T, er ekvivalent med

$$\forall \mathbf{h} : 0 < ||\mathbf{h}||_1 < \delta, \mathbf{h}\cdot\mathbf{z} \mod N \neq 0 \tag{15}$$

Dersom ein varierer $\mathbf{h}$ systematisk i eit område for $||\mathbf{h}||_1 = \delta^{'} > 0$ for $\delta^{'}$ aukande og testar for kravet i (15), finn ein $\delta = \delta^{'} - 1$ fyrste staden testen feilar. Algoritmen vil bli skildra detaljert i kapittelet om koding og implementering, og er basert på ei algoritme gitt av Ronald Cools [5].

3.3 Deltasekvensar

Alt eg har presentert no er standard stoff for latticereglar og finnast i litteratur. For å kunne gå vidare med å finne gode $\mathbf{z}$ vektorar treng vi noko meir teori. Teorien eg no vil presentere er tidlegare upublisert materiale frå eit sett av notat skrive av James Lyness [28] [15]. Desse notatane inneheld mellom anna nokre svært nyttige teorem som er til hjelp i algoritmer for konstruksjon av gode latticeregla, og fortel korleis $\mathbf{z}$ må vere oppbygd for å generere latticereglar av gitt trigonometrisk grad.

Vektorane $\mathbf{z}$ kan sjåast på som talfylgjer, og ut i frå teorem 2.2 kan ein utan problem sortere komponentane i stigande rekkjefylgje. Dette vil berre korrespondere til ei renumerering av dimensjonane og fører ikkje til noko tap av generalitet. For ein latticeregel med utvida grad δ kan ein vise at om vi krev at

$$\mathbf{z} = (z_1, \ldots, z_i, \ldots, z_s); z_i = 1, 1 \leq i \leq s \tag{16}$$

må den kunne konstruerast ut i frå ein deltasekvens. Vi kan også vise ut i frå dette at for ein kvar δ må det finnast ein latticeregel av utvida grad δ generert ut i frå ein $\mathbf{z}$ på denne forma.

Definisjon 3.2 (Deltasekvens)**.** Ein vektor $\mathbf{a} = (a_1, \ldots, a_\sigma)^T$ vert kalla deltasekvens dersom

$$0 < a_1 < \cdots < a_\sigma \tag{17}$$

$$|\sum_{j=1}^{\sigma} \lambda_j a_j| + \sum_{j=1}^{\sigma} |\lambda_j| \geq \delta \text{ for alle } \lambda \in \mathbb{Z}^\sigma. \tag{18}$$

Dersom ein har ein λ som gjer (18) til ein likskap vert $\mathbf{a}$ kalla ein strikt deltasekvens.

Vi kan no kome med eit viktig teorem for denne oppgåva.

Teorem 3.1 (Deltasekvens)**.** *Eitkvart rank 1 lattice Λ av utvida grad δ generert av $\mathbf{z} \in \mathbb{Z}^s$ og N, der $\mathbf{z} = (z_1, \ldots, z_{i-1}, 1, z_{i+1}, \ldots, z_s)^T; 1 \leq i \leq s$, må vere i same ekvivalensklasse som eit lattice Λ' generert av $\mathbf{z}'$ og N der $\mathbf{z}'$ er danna av ein deltasekvens av grad δ.*

Prov For ein $\mathbf{z}' = (z_1', \ldots, z_i', \ldots, z_s')^T$ der $z_i' = 1, 1 \leq i \leq s$ kan vi fyrst, ut i frå Teorem 2.2 sortere alle elementa i $\mathbf{z}'$ i stigande rekkjefylgje slik at $\mathbf{z} = (1, z_2, \ldots, z_s)^T$ Dette gir oss at for einkvar $\mu \in \mathbb{Z}^s$ vil alle $\mathbf{h}$ på forma

$$\mathbf{h} = [(N\mu_1 - \sum_{i=2}^{s} \mu_i z_i), \mu_2, \ldots, \mu_s]^T$$

vere punkt i $\Lambda^\perp$. Vi kan også skrive dette som

$$\mathbf{h} = [(N\mu_1 + \sum_{i=2}^{s} \mu_i z_i), -\mu_2, \ldots, -\mu_s]^T. \tag{19}$$

Ut i frå dette fyl det av (15) at $||\mathbf{h}||_1 \geq \delta$ for ein latticeregel av utvida grad δ. For ein vilkårleg μ får vi då at

$$||\mathbf{h}||_1 = ||[(N\mu_1 + \sum_{i=2}^{s} \mu_i z_i), -\mu_2, \ldots, -\mu_s]^T||_1$$

$$||\mathbf{h}||_1 = |\sum_{j=2}^{s} \mu_j z_j + \mu_1 N| + \sum_{j=2}^{s} |\mu_j|$$

$$\Updownarrow$$

$$|\sum_{j=2}^{s} \mu_j z_j + \mu_1 N| + \sum_{j=2}^{s} |\mu_j| \geq \delta \tag{20}$$

Dette må særskild gjelde når $\mu_1 = 0$. Vi ser ut i frå dette at om vi set $\mathbf{a} = (z_2, \ldots, z_s)^T$ med $\sigma = s - 1$ element og $\lambda = (\mu_2, \ldots, \mu_s)^T$, får vi at dersom $\mathbf{z}$ skal generere ein latticeregel av utvida grad δ må $\mathbf{a}$ vere ein deltasekvens av grad δ . ■

Eit resultat av dette og Teorem 2.3 er at vi for einkvar δ alltid kan finne ein latticeregel av rank 1 og utvida grad δ generert av ein $\mathbf{z} = (1, z_2, \ldots, z_s)$ der $0 < z_1 = 1 < z_2 < \cdots < z_s < N/2$.

Vi ser at for latticereglar Λ av denne typen og for $\mathbf{h} \in \Lambda^\perp$ finnast det ein $\mu \in \mathbb{Z}^\sigma$ slik at

$$||\mathbf{h}||_1 = |N\mu_1 + \lambda \cdot \mathbf{a}| + ||\lambda||_1.$$

Vi ser at $||\mathbf{h}||_1 \geq ||\lambda||_1$, så ved å undersøkje alle $\mathbf{h}$ generert av $\mu = (0, \lambda)^t$ der $||\lambda||_1 \leq \delta$ har vi garantert fått med alle $\mathbf{h} \in \Lambda^\perp$ som trengst for å evaluere (15). Vi let m vere talet på $\mathbf{h}$ vi må evaluere for kvart par av $\mathbf{z}$ og N.

Dette leier naturleg til ein 2-stegs strategi for søk etter latticereglar. Fyrste steg vil vere å finne gode deltasekvensar, $\mathbf{a}$, før ein i andre steg finn ein minst mogleg N slik at $\mathbf{z}$ og N genererer eit lattice av utvida grad δ. Frå teorem 2.3 ser vi at z_s er ei nedre grense for $N/2$, og det vil derfor vere naturleg å finne deltasekvensar med låg verdi på z_s. I resten av dette kapittelet vil eg konsentrere meg om teori for steg ein.

3.4 Krav for deltasekvensar

For å kunne søkje etter deltasekvensar effektivt må vi vite noko om eigenskapane deira. Vi må setje nokre krav til $\mathbf{z}$ for at den skal vere ein deltasekvens.

Lemma 3.1. *Nødvendige krav for at ein sekvens* $\mathbf{z}$ *skal vere ein deltasekvens av utvida grad større eller lik* δ *er*

$$z_1 \geq \delta - 1$$

$$z_j - z_{j-1} \geq \delta - 2; 1 < j \leq s$$

Prov Begge desse kan lett visast ved motseiing. Vi antar at $z_1 \leq \delta - 2$. Dersom då $\lambda = (1, 0, \ldots, 0)$ får vi at uttrykket i (20) reduserer til $z_1 + 1 \leq \delta - 2 + 1$ som er mindre enn δ. Vi må derfor ha at $z_1 \geq \delta - 1$.

Tilsvarande, dersom vi antar at det for $\mathbf{z}$ finnast eit par $z_j - z_{j-1} \leq \delta - 3$ får vi då for $\lambda_j = -\lambda_{j-1}$ og med 0 elles at (20) reduserer til $\lambda_j z_j - \lambda_{j-1} z_{j-1} + 2 \leq \delta - 3 + 2 < \delta$. ■

Teorem 3.2 (Reduksjon og ekspansjon av deltasekvensar)**.** *Dersom* $\mathbf{a} = (a_1, a_2, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_s)$ *er ein deltasekvens så vil også* $\tilde{\mathbf{a}} = (a_1, a_2, \ldots, a_{i-1}, a_{i+1}, \ldots, a_s)$ *vere ein deltasekvens.*

Dersom $\mathbf{a} = (a_1, a_2, \ldots, a_{s-1}, a_s)$ *er ein deltasekvens så må det også finnast ein* $a_{s+1} \geq a_s + \delta - 2$ *som gjer at* $\bar{\mathbf{a}} = (a_1, a_2, \ldots, a_s, a_{s+1})$ *blir ein deltasekvens.*

Prov Dersom $\tilde{\mathbf{a}}$ ikkje er ein deltasekvens må det finnast ein $\tilde{\lambda} = (\lambda_1, \ldots, \lambda_{i-1}, \lambda_{i+1}, \ldots, \lambda_s)$ som gjer at kravet (18) ikkje held. For $\mathbf{a}$ må også den tilsvarande $\lambda = (\lambda_1, \ldots, \lambda_{i-1}, \lambda_i, \lambda_{i+1}, \ldots, \lambda_s)$ med $\lambda_i = 0$ gjere at (18) ikkje held. Dette fører til ei sjølvmotseiing og $\tilde{\mathbf{a}}$ må derfor også vere ein deltasekvens.

Dersom $\mathbf{a}$ er ein deltasekvens har vi at (18) vil gjelde for alle $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_s)$ og dermed også for $\bar{\mathbf{a}}$ og $\bar{\lambda} = (\lambda, 0)$. For $\lambda_{s+1} \neq 0$ treng vi kun å konsentrere oss om λ der $|\lambda|_1 < \delta - 1$. Då vil

$$a_{s+1} > (\delta - 2)a_s$$

alltid gi ein ny deltasekvens $\bar{\mathbf{a}}$. På grunn av (3.1) kan a_{s+1} ikkje vere mindre enn $a_s+\delta-2$. ■

Som eg har nemnd vil eg i denne oppgåva konsentrere meg om $\delta = 5$, og eg vil derfor kome med ei teorem, Teorem 3.3, som er viktig for nettopp tilfellet $\delta = 5$. Men før det må vi få på plass to lemma.

Lemma 3.2. *La* $0 < a_1 < a_2 < \cdots < a_s$, $2a_1 - a_s \geq 0$, $\sum_{i=1}^{s} |\lambda_i| \leq 4$ *og* $\sum_{i=1}^{s} \lambda_i \neq 0$. *Då vil*

$$S = |\sum_{i=1}^{s} \lambda_i a_i| \geq 2a_1 - a_s.$$

Vi viser dette ved å sjå på alle moglegheitene for å konstruere λ ut i frå desse krava.
For $\lambda_1 \neq 0$ og $\lambda_i = 0 \forall\, i \neq 1$ har vi $S \geq |\lambda_1|a_1 \geq 2a_1 - a_s$.
For $\lambda_1 = \{2, 3\}; \lambda_i = -1 \forall\, i > 1$ blir $S = |\lambda_1 a_1 - a_i| \geq 2a_1 - a_s$.
For $\lambda_1 = 2, \lambda_i = -\lambda_j = 1$ får vi $S = |2a_1 + a_i - a_j| \geq 2a_1 - a_s$
For $\lambda_1 = \lambda_i = -\lambda_j = 1$ får vi $S = |a_1 + a_i - a_j| \geq 2a_1 - a_s$
For alle andre moglegheiter der $\sum_{i=1}^{s} \lambda_i \neq 0$ har vi at forteikna til alle $\lambda_i \in \lambda; \lambda_i \neq 0$ er like. Dette gir $S \geq a_i \geq a_1$. ■

Lemma 3.3. *Når* $2a_1 - a_s \geq \delta - 1$ *vert kravet for at* $\mathbf{a} = (a_1, a_2, \ldots, a_s)$ *skal vere ein deltasekvens at*

$$0 < \sum_{i=1}^{s} |\lambda_i| \leq \delta - 1 \text{ og } \sum_{i=1}^{s} \lambda_i = 0$$

Prov Dette ser vi ut i frå at for $\sum_{i=1}^{s} \lambda_i \neq 0$ får vi $\sum_{i=1}^{s} |\lambda_i| + |S| \geq \sum_{i=1}^{s} |\lambda_i| + 2a_1 - a_s \geq 1 + \delta - 1$. ■

Teorem 3.3 (Tilstrekkjelege og nødvendige krav til deltasekvensar av utvida grad 5)**.** *La* $a_s < 2a_1 - \delta + 1$ *og* $\epsilon_{i,j} = |a_i - a_j|$. *Då vil* $\mathbf{a} = (a_1, a_2, \cdots, a_s)$ *danne ein deltasekvens med* $\delta \geq 5$ *dersom, og kun dersom*

$$E = \{\epsilon_{i,j}; s \geq i > j \geq 1\}$$

inneheld distinkte element og $\epsilon_{i,j} \geq 3$

Prov Fyrst vil vi vise at dersom $\mathbf{a}$ skal vere ein deltasekvens må E innehalde distinkte element.

Dersom

$$\epsilon_{i,j} = \epsilon_{k,l}$$

då er

$$a_i - a_j - a_k + a_l = 0.$$

Dersom vi så vel

$$\lambda_i = -\lambda_j = -\lambda_k = \lambda_l = 1; \lambda_r = 0 \forall r \notin i, j, k, l$$

får vi

$$\sum_{m=1}^{s} |\lambda_m| + |\sum_{m=1}^{s} \lambda_m a_m| = 4 < 5.$$

Dette viser at fyrste kravet er nødvendig for at $\mathbf{a}$ skal vere ein deltasekvens.

For å vise at dette er eit tilstrekkjeleg krav treng vi, ut i frå lemma 3.3, berre å vurdere dei tilfella der $\sum_{i=1}^{s} \lambda_i = 0$.

For $\lambda_i = -\lambda_j$ får vi $S = |\lambda_i(a_i - a_j)|$ som gir $\sum_{k=1}^{s} |\lambda_k| + S \geq 2 + \epsilon_{i,j} \geq \delta$

For $\lambda_i = -\lambda_j = \lambda_k = -\lambda_l$ får vi $S = | \pm \epsilon_{i,j} \pm \epsilon_{k,l}|$. Sidan alle ϵ er distinkte får vi $\sum_{m=1}^{s} |\lambda_m| + S \geq 4 + S \geq \delta$

For $\lambda_i = -2\lambda_j = -2\lambda_k$ blir $S = |2a_i - a_j - a_k| = |\pm\epsilon_{i,j} \pm \epsilon_{j,k}|$. Dette gir $\sum_{l=1}^{s} |\lambda_l| + S \geq 4 + S \geq \delta$. ■

3.5 Golomblinjalar og kopling til deltasekvensar

Kravet om distinkte avstandar kjenner vi att som det som definerer eit anna fenomen kalla golomblinjalar.

Definisjon 3.3 (Golomblinjal)**.** Ein vektor, $\mathbf{a} = (a_1, a_2, \ldots, a_\sigma)$; $a_1 < a_2 < \cdots < a_\sigma$, der alle tala er heiltal, vert kalla ein golomblinjal dersom det for eit heiltal, $x \neq 0$, finnast høgst ei løysing på likninga $x = a_j - a_i$, a_i, a_j.

Eit enkelt tal i ein slik golomblinjal vert kalla eit merke. Lengda av ein golomblinjal $G(\sigma)$ vert definert som $G(\sigma) = a_\sigma$ der a_σ er det merke med høgast verdi. Ein optimal golomblinjal er den kortast moglege for eit gitt tal på merker.

To operasjonar som er nyttige, og som gjer at golomblinjalane framleis er golomblinjal, er fjerning av merker og forskuving av merka med ein heiltalsverdi.

Teorem 3.4 (Forskuving og trunkering av golomblinjal)**.** *For einkvar golomblinjal* $\mathbf{a} = (a_1, \ldots, a_{\sigma-1}, a_\sigma)$ *og konstant* $c \in \mathbb{Z}$ *vil også* $\mathbf{a}' = (a_1, \ldots, a_{\sigma-1})$ *og* $\mathbf{a}'' = (a_1 + c, \ldots, a_{\sigma-1} + c, a_\sigma + c)$ *vere golomblinjal.*

Prov Dersom vi har ein golomlbinjal $\mathbf{a} = (a_1, a_2, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_\sigma)$ kan vi fjerne eit merke og får $\mathbf{a}' = (a_1, a_2, \ldots, a_{i-1}, a_{i+1}, \ldots, a_\sigma)$. Dersom $\mathbf{a}'$ ikkje er golomblinjal må det finnast j, k, l, m slik at $a_j - a_k = a_l - a_m$. Men sidan operasjonen vi har utført er å fjerne eit merke, må desse merka framleis finnast i $\mathbf{a}$, som impliserer at $\mathbf{a}$ ikkje er golomblinjal. Ved motseiing må derfor også $\mathbf{a}'$ vere golomblinjal.

For å vise at forskuving gir ein ny golomblinjal antar vi fyrst at $\mathbf{a} = (a_1, a_2, \ldots, a_\sigma)$ er golomblinjal. For ein $c \in \mathbb{Z}$ har vi $\mathbf{a}'' = (a_1 + c, a_2 + c, \ldots, a_\sigma + c)$. Dersom $\mathbf{a}''$ ikkje er golomblinjal må det finnast i, j, k, l slik at

$$a_i + c - (a_j + c) = a_k + c - (a_l + c)$$

$$\Downarrow$$

$$a_i - a_j = a_k - a_l.$$

Men dette impliserer at $\mathbf{a}$ ikkje er golomblinjal og vi får ei motseiing. Vi får då at $\mathbf{a}''$ også må vere golomblinjal. ■

Golomblinjalar dukka fyrst opp i eit arbeid av Wallace C. Babcock frå 1953 [2] som ei løysing på eit problem i samband med radioteknikk. Ein kunne redusere interferens

mellom ulike frekvensar ved å justere dei ulike kanalane ut i frå merka på ein golomblinjal. Golomblinjalen er oppkalla etter Solomon W. Golomb som var den fyrste til å studere dei systematisk.

Ein ekvivalent definisjon er sidonsett, som definerast ut i frå at det inneheld distinkte summar for kvart par av merker. Vi ser lett denne ekvivalensen ved at om $a_i - a_j \neq a_k - a_l$, $\forall \{i, j, k, l\} \in \mathbb{Z}$ må også $a_i + a_l \neq a_k + a_j$. Sidonsett har blitt studert sidan 1930 talet, og vart fyrst skildra av Simon Sidon [21], saman med Pál Erdős og Pál Turán [8]. Frå teori bak sidonsett har vi ei nedre grense for lengda på ein golomblinjal [14]

$$G(\sigma) \geq \sigma^2 - 2\sigma\sqrt{\sigma} + \sqrt{\sigma} - 2. \tag{21}$$

Ekvivalens med golomblinjalar har blitt demonstrert av Apostolos Dimitriomanolakis [7].

Ingen lukka løysing på konstruksjon av optimale golomblinjalar finnast endå. Dei vi kjenner er funne ved systematiske søk, fyrst manuellt og seinare med datamaskiner. Særskild kan ein nemne projectOGR drive på www.distributet.net [9], som er eit massivt kooperativt søk etter optimale golomblinjalar.

Det finnast likevel fleire ulike konstruksjonar som gir nær optimale golomblinjalar. Med nær optimale golomblinjalar meinar ein golomblinjalar med lengde $a_\sigma \leq \sigma^2$, som er i nærleiken av denne nedre grensa nemnd over. Dette fyl også eit framlegg, presentert av Erdős [8], som seier at alle optimale golomblinjalar med σ merker er kortare enn σ^2. Dette har ikkje blitt universellt prova, men det har blitt vist korrekt for $s < 65000$ [1] [13] [7].

Ein spesiell måte å framstille golomblinjalar på, som eg har gjort nytte av i dette arbeidet er det som vert kalla sykliske golomblinjalar, eller også modulære golomblinjalar. Dei er kort fortald kjenneteikna ved at dei der ein vanleg golomblinjal er definert ut i frå å ha unike differansar mellom merka, har ein syklisk golomblinjal unike differansar modulo eit heiltal b. Meir persist vert dette definert slik.

Definisjon 3.4 (Sysklisk golomblinjal)**.** For to tal $a_i, a_j \in \mathbf{a}; i \neq j$ der $\mathbf{a}$ er ein syklisk golomblinjal, har vi at $(a_i - a_j) (\text{mod } b) \neq (a_j - a_i) (\text{mod } b)$

Ein nyttig måte å illustrere ein slik golomblinjal er å ta ein sirkel med omkrins b og plassere ut distansane rundt denne sirkelen. Derav har vi namnet sykliske golomblinjalar.

Ein grunn til at sykliske golomblinjalar er av interesse i denne oppgåva er at mange av dei gode konstruksjonane som gir korte golomblinjalar er basert på modulære konstruksjonar og gir nettopp sykliske golomblinjalar. Dei beste av dei gir golomblinjalar med lengde rundt s^2, som også er i nærleiken av denne nedre grensa for lengda av golomblinjalar nemnd over. Dette er praktisk for oppgåva i og med at vi er på jakt etter golomblinjalar som er så korte som mogleg.

Når vi fyrst har funne ein syklisk golomblinjal med s merker, kan vi konstruere $s - 1$ nye golomblinjalar ved forskuving av golomblinjalen modulo b. Det vil seie, om vi har ein golomblinjal $\mathbf{a} = a_1, a_2, \ldots, a_s$, kan vi danne ein ny golomblinjal ved transformasjonen

$$\mathbf{a}' = S_c(\mathbf{a}) = \{a_1 + c (\text{mod } b), a_2 + c (\text{mod } b), \ldots, a_s + c (\text{mod } b)\} \tag{22}$$

der c er eit vilkårleg heiltal. Ved å velje $c = b - a_s$ får vi ein ny golomblinjal der det øverste merket har blitt det lågaste. Denne prosessen kan halde fram inntil vi er tilbake til den originale golomblinjalen, og vi vil ha eit sett på s ulike linjalar. Alle desse vil vere deltasekvensar, men generelt ulike med omsyn til grad.

Som vi såg i Teorem 3.3 må ein deltasekvens $\delta \geq 5$, i tillegg til å vere golomblinjal, ha minste avstand $\min(\epsilon_{i,j}) \geq 3$. Dette gjer at om ein kan konstruere golomblinjalar med dei rette eigenskapane kan ein også, i alle fall ut i frå teorien, konstruere gode latticereglar. Problemet er at ein ikkje for tida har tilstrekkjeleg gode krav til desse golomblinjalane, og ein kan berre eksperimentellt finne gode kandidatar mellom dei mange mindre gode. Det er helder ikkje blitt vist at ein optimal golomblinjal automatisk vil gi ein optimal latticeregel. Samanhangen her er at på grunn av Teorem 2.3 vil vi rekne med at dei beste $\mathbf{z}$ har z_s liten. Dessutan er det ikkje så lett å seie kva som er ein god latticeregel når ein ikkje allereie har den optimale.

Framgongsmåten eg her vil nytte vil vere å fyrst konstruere ein nær optimal golomblinjal, for så å fjerne eit eller to merker slik at $\min(\epsilon_{i,j}) \geq 3$. Vi kan kalle mengda av alle golomblinjalar med minste avstand $\epsilon i, j \geq k$ for GL_k, og vi er derfor ute etter deltasekvensar $\mathbf{a} \in GL_3$. Implementeringa og algoritmene frå denne teorien vil eg skildre i neste kapittel.

4 Algoritmer og kompleksitet

4.1 Algoritmer

For no å kunne arbeide vidare må vi kunne implementere denne teorien i robuste algoritmer. Dei algoritmene vi treng for å gjennomføre vårt søk etter gode latticereglar består fyrst og fremst ein metode for å rekne ut trigonometrisk grad. Denne, som vi skal sjå, kan delast i to mindre algoritmer. Vidare vil vi ha behov for å kunne konstruere $\mathbf{z}$, og ut i frå førre kapittel, også metodar for å konstruere deltasekvensar. Dette er også ein todelt prosess, der vi fyrst dannar deltasekvensar, for så å modifisere dei på ulike måtar for å gi ein $\mathbf{z}$ av ynska grad for gitt dimensjon. Til slutt treng vi ei algoritme for å setje saman desse til eit effektivt søk.

Programmet eg har nytta baserer seg på å konstruere latticereglar ved hjelp av golomblinjalar, for så å teste latticereglane for trigonometrisk grad. Det er likevel ingenting i vegen for å nytte algoritmene for testing av trigonometrisk grad i eit program som tek ikkje eksplisitt sykliske golomblinjalar som input. Det at golomblinjalane er sykliske i mitt program gjer det enklare å manipulere dei, slik at vi kan teste fleire linjalar ut i frå den same konstruksjonen. Dessutan dannar alle dei beste av dei kjende konstruksjonane sykliske golomblinjalar. Vi vil derfor gjerne også utnytte denne eigenskapen i programmet.

Andre ting vi vil utnytte i denne oppgåva er det faktum at det finnast fleire ulike metodar for å generere golomblinjalar. Dette gir grunnlag for ein del heuristiske metodar for å forbetre søket etter latticereglar. Eg vil kommentere dette for dei algoritmene der dette er aktuelt.

4.2 Utrekning av trigonometrisk grad

Før eg no gjer noko anna vil eg presentere metoden eg brukar for å finne trigonometrisk grad. Standardmåten å rekne ut trigonometrisk grad er å variere $\mathbf{h}$ med $||\mathbf{h}||_1 = d$ for stigande d, for så å teste $\mathbf{h} \cdot \mathbf{z} (\text{mod } N) \neq 0$ inntil dette feilar **??**h-norm<delta). For den $\mathbf{h}$ der testen feilar har vi $\delta = ||\mathbf{h}||_1$. For mine algoritmer har eg teke utgangspunkt i eit program skildra av Ronald Cools [5], som nettopp er basert på denne opplagde algoritmen. Dersom vi kun er ute etter å vite om ein gitt latticeregel har utvida grad $\delta \geq \hat{\delta}$, og ikkje treng å vite den eksakte grada til latticeregelen, kan vi stoppe når alle $\mathbf{h}$ der $||\mathbf{h}||_1 < \hat{\delta}$ er evaluert.

Eit problem med denne metoden å gjere det på er at når vi vil undersøke fleire latticereglar har vi behov for å generere mange $\mathbf{h}$ vektorar. Utfordringa vert då å generere kun dei $\mathbf{h}$ vektorane vi har behov for, og ikkje fleire.

Det nye eg har her i mine algoritmer er ein observasjon som gjer at vi kan dele prosessen med å finne trigonometrisk grad inn i to trinn, og i tillegg kune ta vare på ein del informasjon utan å måtte rekne ut alt på nytt for kvar nye N. Dersom vi vil evaluere $\mathbf{h} \cdot \mathbf{h}$ for mange $\mathbf{z}$ kan det vere nyttig å lagre alle $\mathbf{h}$ i ei matrise H. I tillegg, om vi så vil evaluere $Hz (\text{mod } N)$ for mange N kan det vere lønsamt å lagre $\mathbf{q} = H\mathbf{z}$ og helder evaluere $\mathbf{q} (\text{mod } N)$.

I mine algoritmer vil eg også i hovudsak nytte denne metoden for å forbetre køyretida. Eg vil fyrst leggje fram algoritmen finn**q**(), som dannar kjernen for desse utrekningane.

Algorithm 1 finn$\mathbf{q}(\mathbf{z}, \hat{\delta})$

$d \leftarrow 1$ {Inneverande trigonometrisk grad}
$\mathbf{h} \leftarrow (1, 0, \ldots, 0)$
$\mathbf{q} \leftarrow (0, \ldots, 0)$ { Lengde lik $\sum_{i=1}^{\hat{\delta}-1} 2^{i-1} \begin{pmatrix} s \\ i \end{pmatrix} \begin{pmatrix} \hat{\delta}-1 \\ i \end{pmatrix}$}
$\Delta\mathbf{q} \leftarrow \mathbf{q}$
$k \leftarrow 0$
for $d \leq \hat{\delta}$ **do**
  for $\forall \mathbf{h}; ||\mathbf{h}||_1 = d$ **do**
    $k++$
    $\mathbf{q}_k \leftarrow \mathbf{h} \cdot \mathbf{z}$
    $\Delta\mathbf{q}_k = \sum_{i=1}^{d} \text{ sign}(\mathbf{h}_i)$
  end for
end for
return $\mathbf{q}, \Delta\mathbf{q}$

På grunn av at $\mathbf{h}$ er glissen når δ er lågare enn s kan vi utnytte dette til å spare endå meir tid og lagringsplass. Det finnast allereie gode algoritmer for å rekne med glisne vektorar og matriser, og eg vil derfor ikkje gå inn på desse teknikkane her.

Legg merke til at det er ein skilnad på denne algoritmen og teorien over. I denne algoritmen lagrar eg ikkje H matrisen, men helder vektoren $\mathbf{q}$ og ein annan vektor $\Delta\mathbf{q}$ som representerer endringa i $\mathbf{q}$ som resultat av endringa $\Delta\mathbf{z}$ i $\mathbf{z}$. Grunnen til dette er at alle $\mathbf{z}$ vi evaluerer er danna ved å konstruere deltasekvensar ut i frå golomblinjalar, og som vi vil sjå under, treng vi kun å rekne ut $H\mathbf{z}$ ein gong for kvar deltasekvens konstruert. Valet er gjort ut i frå eit omsyn både til tidsbruken og til kravet for lagringsplass. Å lagre heile H kan vere fornuftig dersom ein gjennomfører eit reint brute force søk, og vil evaluere $H\mathbf{z}$ for kvar $\mathbf{z}$ danna, men når utrekningane uansett vert utført relativt få gongar kan det vere lurt å også spare på lagringsplass.

Frå Teorem 3.1 fyl det at om vi har ein deltasekvens $\mathbf{a}$ av grad δ vil einkvar $\mathbf{z} = (1, \mathbf{a})^T$ saman med ein godt vald N generere ein latticeregel av utvida grad δ. Vi søkjer systematisk etter den beste $\mathbf{z}$ mellom alle danna ut i frå $\mathbf{a}$. Teorem 3.3 viser oss at alle deltasekvensar $\mathbf{a}$ må vere golomblinjalar, og at alle golomblinjalar med minste distanse $\epsilon_{i,j}$ mellom to merker må vere deltasekvens. I tillegg kan vi ut i frå Teorem 3.4, om vi har ein golomblinjal $\mathbf{a}$, danne eit heilt sett av nye golomblinjalar ved å forskuve $\mathbf{a}$. Kvar av desse nye golomblinjalane vil også vere deltasekvensar. Forskuving av $\mathbf{a}$ kan vi bruke for å utvide søkjeromet utan å måtte rekne ut $H\mathbf{z}$ fleire gonger, noko vi vil demonstrere under. Å forskuve $\mathbf{a}$ med ein konstant distanse tilsvarar å endre på $\mathbf{z}$ med ein $\Delta\mathbf{z}$. Dei $\Delta\mathbf{z}$ eg vil bruke er $\Delta\hat{\mathbf{z}} = (1, \ldots, 1)^T$ og $\Delta\mathbf{z} = (0, 1, \ldots, 1)^T$. Grunngjevinga for å bruke akkurat desse $\Delta\mathbf{z}$ vil vi sjå i kapittelet om algoritmer for konstruksjon av deltasekvensar, som kjem under her. Vi får då for kvar $\mathbf{z}_1 = \mathbf{z}_0 + \Delta\mathbf{z}$ likninga

$$\begin{aligned} \mathbf{q} &= H\mathbf{z}_1 \\ &= H(\mathbf{z}_0 + \Delta\mathbf{z}) \end{aligned}$$

$$= H\mathbf{z}_0 + H\Delta\mathbf{z}. \tag{23}$$

Ved å la $\Delta\mathbf{q} = H\Delta\mathbf{z}$ og la $\mathbf{q}_{i+1} = \mathbf{q}_i + \Delta\mathbf{q}$ sparer vi endå meir utrekningar, og det er dette siste trikset som gjer at vi verkeleg får ned køyretida samanlikna med brute force metoden. Ved å bruke denne utrekninga treng vi helder ikkje å lagre heile H matrisen, men vi bygg opp $\mathbf{q}$ og $\Delta\mathbf{a}$ som i linje 9 og 10. Dette verkar kun dersom vi nyttar deltasekvensar beslekta ved forskuving. Vi har med dette ein algoritme for fyrste trinn i utrekning av grad. Siste trinn, testing for aukande N, inngår i algoritmen testLatticeregel() under, men fyrst vil eg sjå på algoritmer for å konstruere deltasekvensar.

4.3 Algoritmer for konstruksjon av deltasekvensar

Frå Teorem 3.3 har vi at alle deltasekvensar av grad $\delta = 5$ må vere golomblinjalar med alle avstandar $\epsilon_{i,j} \geq 3$. Samstundes garanterer dette teoremet at om vi har ein golomblinjal $\mathbf{a} \in GL_3$ vil den også vere deltasekvens av grad $\delta = 5$. Dette motiverer oss til å konstruere $\mathbf{z}$ ut i frå denne teorien.

Det finnast mange ulike måtar å konstruere golomblinjalar på, men Teorem 2.3 gir oss grunn til å sjå etter metoder for å danne golomblinjalar så korte som mogleg. Som nemnt finnast det ein teori om at for einkvar $\sigma \in \mathbb{Z}$ finnast det ein golomblinjal med σ merker og med lengde $G(\sigma) \leq \sigma^2$, og alle golomblinjalar kortare enn dette vert kalla nær optimale golomblinjalar. To konstruksjonar som gir golomblinjalar med lengde i nærleiken av σ^2 er Ruzsa konstruksjonen [20] og Bose-Chowla konstruksjonen [3], og dei har begge blitt brukt i eit arbeid av Dimitriomanolakis [7] for å konstruere golomblinjalar opp til 65000 merker og med lengde kortare enn σ^2. Dei vil derfor kunne gi golomblinjalar som er nær optimale, og sidan vi er ute ettere så korte linjalar som mogleg vil eg bruke desse konstruksjonane i mi oppgåve.

Ruzsakonstruksjonen baserer seg på å løyse kongruensen

$$a_i \equiv i \cdot \mathrm{mod}(p-1); a_i \equiv g^i \mathrm{mod}(p),$$

der p er eit primtal og g er eit primitiv rot modulo p, og $i = \{1, 2, \cdots, p-1\}$. Dette gir ei løysing

$$a_i = [pi + (p-1)g^i] \mathrm{mod}(p(p-1))$$

opp til ein kongruens. Resultatet blir ein golomblinjal med $p-1$ merker, a_i, og distinkte avstandar modulo $p(p-1)$

Algorithm 2 konstruerSykliskGL(p) Ruzsa konstruksjon

1: $s \leftarrow p - 1$
2: $c \leftarrow p$
3: $g \leftarrow$ primitivt rot modulo p
4: $d \leftarrow (p-1)g$
5: $b \leftarrow p(p-1)$
6: $\mathbf{z} \leftarrow \mathbf{0}$
7: **for** i=0 **to** s-1 **do**
8: $\quad a_i \leftarrow (c+d) (\text{mod } b)$
9: $\quad c \leftarrow c + p$
10: $\quad d \leftarrow (d \cdot g) (\text{mod } b)$
11: **end for**
12: sorter($\mathbf{a}$)
13: reduser($\mathbf{a}$)
14: **return** $\mathbf{a}$

Bose-Chowla konstruksjonen baserer seg på evaluering av Galois field $GF(p)$ for p primtal. La p vere eit primtal og θ vere eit primitivt element i $GF(p^2)$. Då har dei p heiltala

$$a_1, a_2, \ldots, a_p = \alpha : 1 \leq \alpha < p^2 \,\&\, \theta^\alpha - \theta \in GF(p)$$

parvist distinkte avstandar modulo $p^2 - 1$.

Ein heldig eigenskap hos denne konstruksjonen er at alle a_i vert danna i stigande rekkjefylgje etter storleik, og trengst derfor ikkje å sorterast, som er tilfellet med med Ruzsa konstruksjonen. Det er lett å sjå at $\max a_p < p^2 - 1$ for $\mathbf{a}$ normalisert til $a_1 = 0$.

Algorithm 3 konstruerSykliskGL(p) Bose-Chowla konstruksjon

$f \leftarrow$ irredusibelt element i $GF(p)$ av orden 2
$\zeta \leftarrow \theta \leftarrow$ primitivt element i $GF(p)$ av orden 1
for $n = 1$ **to** p^2 **do**
$\quad$ **if** $\zeta - \theta \in GF(p)$ **then**
$\quad\quad z[i] \leftarrow n$
$\quad\quad i++$
$\quad$ **end if**
$\quad \zeta \leftarrow \zeta \cdot \theta (\text{mod } f)$
end for
return z

Ved hjelp av desse to konstruksjonane kan vi danne $\mathbf{z}$ vektorar som tilfredsstiller kravet i Teorem 3.1. Ser vi på likning (16) og Teorem 3.3 vil det vere naturleg å la $\mathbf{z} = (1, \mathbf{a})^T$ for $\mathbf{a}$ deltasekvens der $z_i = a_{i-1}$; $\forall 1 < i \leq s$, $z_1 = 1$, og $\mathbf{z}$ vil ha $s = \sigma + 1$ dimensjonar.

Desse konstruksjonane gir golomblinjalar kun for kvart primtal p. Dette gjer at det blir nokre hol mellom linjalane konstruert på denne måten. Desse hola kan fyllast ut

ved å trunkere $\mathbf{z}$. Vi har mange alternativ for korleis ein vil trunkere $\mathbf{z}$, men ut i frå ynskjet om å få så korte $\mathbf{z}$ som mogleg vil det vere naturleg å fjerne z_s for å danne ein ny $\mathbf{z}$ av dimensjon $s-1$. Dette vil vere analogt med fyrst å konstruere deltasekvensar $\mathbf{a} = (a_1, a_2, \ldots, a_{\sigma-1}, a_\sigma)^T$ for så å forkorte dei. Dette gjer vi ved å fjerne anten a_σ eller a_1 før ein dannar $\mathbf{z}$. Men for sykliske golomblinjalar vil desse to operasjonane danne eit identisk sett av nye golomblinjalar, og vi treng berre å nytte ein av dei.

Eit alternativ til dette, som eksperimentelt har vist seg å vere like bra, om ikkje betre, er å la fyrste trunkering vere å ta bort $z_1 = 1$. Dette vil vere analogt med å la $\mathbf{z} = \mathbf{a}$, og eg vil behandle det på denne måten frå no av. Grunnen til at dette kan vere interessant ser vi frå Teorem 2.2. Dersom ein god $N = z_i \pm 1$ finnast som gir grad $\delta \geq 5$, kan vi ved den prosessen skildra i provet for teoremet finne ein $\mathbf{z}' = (1, z_2', \ldots, z_s')^T$ i same ekvivalensklasse som $\mathbf{z}$. I dei tilfella der vi har funne ein god N og vi ikkje kan finne ein $(1, z_2', \ldots, z_s')^T$ ved dene prosessen kan vi likevel rekne med å finne ein

$$\mathbf{z}'' = (1, z_2'', \ldots, z_s'') = c\mathbf{z} (\text{mod } N),\ c \in \mathbb{Z} \tag{24}$$

der $z_i \neq 0 \forall 1 \leq i \leq s$, for eit godt val av c og N, og som også genererer Λ.

Mange fleire alternativ finnast i tillegg til desse, men på grunn av avgrensa tid på oppgåva har eg valt å bruke berre desse to metodane.

Desse to alternativa vil også ha noko å seie for valet av $\Delta\mathbf{z}$ i likning (23). Dersom Vi vel å la $\mathbf{z} = (1, \mathbf{a})^T$ vil vi ha $\Delta\mathbf{z} = (0, 1, \ldots, 1)^T$, slik at $z_1 = 1$ for alle $\mathbf{z}'$ danna ved $\mathbf{z} + c\Delta\mathbf{z}$. Dersom vi let $\mathbf{z} = \mathbf{a}$ vil vi få $\Delta\mathbf{z} = (1, \ldots, 1)^T$.

Til saman får vi dermed fire ulike metodar for å danne $\mathbf{z}$. Metodane r1 og r2 vil heretter vere basert på å danne golomblinjalar $\mathbf{a}$ ved hjelp av Ruzsa konstruksjon, og høvesvist la $\mathbf{z} = (1, \mathbf{a})^T$ og $\mathbf{z} = \mathbf{a}$. Tilsvarande har vi to metodar, heretter kalla b1 og b2, som er basert på å konstruere golomblinjalar ved Bose konstruksjon, og med tilsvarande $\mathbf{z}$ vektorar.

Både Ruzsa og Bose konstruksjonen gir golomblinjalar som er eksplisitt sykliske. Som nemnt i innleiinga, kan vi bruke dette for å utvide søkjeromet etter gode latticereglar. Dersom vi har ein σ dimensjonal $\mathbf{a}$ syklisk golomblinjal i GL_3 normalisert slik at $a_1 = 0$, kan vi ved hjelp av modulær translasjon, som skildra i (22), danne σ nye golomblinjalar. Vi har $\mathbf{a} = (a_1, a_2, \ldots, a_\sigma)^T$ golomblinjal med distinkte avstandar modulo b. La $c = b - a_\sigma$. Set så

$$\mathbf{a}' = (a_1 + c, a_2 + c, \ldots, a_{\sigma-1} + c, a_\sigma + c) (\text{mod } b)$$

$$\mathbf{a}' = (a_1 + c, a_2 + c, \ldots, a_{\sigma-1} + c, b) (\text{mod } b).$$

Om vi sorterer alle a_i' i stigande rekkjefylgje får vi

$$\mathbf{a}' = (0, a_1 + c, a_2 + c, \ldots, a_{\sigma-1} + c)$$

som er ein ny golomblinjal med distinkte avstandar modulo b. Dersom vi utføre denne operasjonen $\sigma - 1$ gongar får vi σ ulike golomblinjalar, medrekna den fyrste $\mathbf{a}$.

Ein bieffekt av dette at vi enkelt kan dele opp søket i fleire mindre trådar, noko som i seg sjølv legg til rette for parallelprogrammering. Ved å la evalueringa av kvar $\mathbf{z}$ danna ut i frå desse golomblinjalane bli utført som ein tråd kvar parallelt.

4.4 Algoritme for søk etter latticeregel

Til slutt vil vi setje saman algoritmen testLatticeregel(p, s, N_{min}, N_{max}). Denne tek fire heiltal, p, s, N_{min}, N_{max} som input, der p er eit primtal og basis for algoritmen for å danne golomblinjalar, s er dimensjon for latticereglane vi vil undersøkje, og N_{min} og N_{max} er minimum og maksimumverdiar for N. Vi må her setje krav til s om at den ikkje kan vere høgare enn ein s_{max} avhengig av konstruksjonsmetoden for $\mathbf{z}$. Algoritmen under er i den form eg har brukt for konstruksjon r1.

Algorithm 4 finnLatticeregel(p, s, N_{min}, N_{max})

1: $\hat{\delta} \leftarrow 5$
2: $\mathbf{a} \leftarrow$ konstruerSykliskGL(p)
3: $N_{god} \leftarrow N_{max}$
4: $b \leftarrow p(p-1)$
5: **for** $k = 1$ **to** s **do**
6: $\quad \mathbf{z} = (1, a_1, \ldots, a_{s-1})^T$
7: $\quad \mathbf{q}, \Delta\mathbf{q} \leftarrow$ finn$\mathbf{q}(\mathbf{z})$
8: $\quad m \leftarrow \mathbf{q}$.length
9: $\quad \mathbf{q} = \mathbf{q} + \Delta\mathbf{q}c$ slik at $q_i \neq 0 \; \forall \; 1 \leq i \leq m$
10: $\quad$ **for** $l = q_s$ **to** $l = N_{god} - 1$ **do**
11: $\quad\quad$ **for** $N = \min(N_{min}, q_s + 2)$ **to** $\max(N_{god}, 4q_s - 2)$ **do**
12: $\quad\quad\quad$ **if** $q_i \bmod N \neq 0 \; \forall \; 1 \leq i \leq m$ **then**
13: $\quad\quad\quad\quad N_{god} \leftarrow N$
14: $\quad\quad\quad\quad \mathbf{z}_{god} \leftarrow (q_1, \ldots, q_s)$
15: $\quad\quad\quad$ **end if**
16: $\quad\quad$ **end for**
17: $\quad\quad \mathbf{q} \leftarrow \mathbf{q} + \Delta\mathbf{q}$
18: $\quad$ **end for**
19: $\quad c \leftarrow b - a_\sigma$
20: $\quad \mathbf{a} \leftarrow (a_1 + c, \ldots, a_\sigma + c) (\bmod b)$
21: $\quad$ sorterElement$(\mathbf{z})$
22: **end for**
23: **return** $\mathbf{z}_{god}, N_{god}$

Loopen i linje 9 til 25 gir kun meining dersom vi har nytta ein eksplisitt syklisk konstruksjon av golomblinjalar. Dersom vi brukar ein generell golomblinjal, vilkårleg av om den er syklisk eller ikkje, vil vi erstatte heile loopen med koden i linje 10 til 21.

Ein bieffekt av at vi lagrar $\mathbf{h} \cdot \mathbf{z}$ i ein vektor $\mathbf{q}$ er at dette kan gjerast slik at $\mathbf{z} = (q_1, \ldots, q_s)^T$, dei s fyrste elementa i $\mathbf{q}$. Dette ser vi lett ut i frå at når $|\mathbf{h}|_1 = 1$ vil $\mathbf{h}_i = \mathbf{e}_i \; \forall \; 1 \leq i \leq s$ der $\mathbf{e}$ er kanonisk einingsvektor,og vi vil få at $q_i = \mathbf{h}_i \cdot \mathbf{z} = z_i$. Dette har vi utnytta i algoritmen over for å sleppe å konstruere alle $\mathbf{z}$ generert ved addisjon med $\Delta\mathbf{z}$. Av tilsvarande argumentasjon kan vi sjå at $\Delta\mathbf{z} = (\Delta q_1, \ldots, \Delta q_s)^T$. Derfor kan vi i linje 16 la $\mathbf{z}_{god} = (q_1, \ldots, q_s)^T$.

Vi har her sett ei øvre grense for N på $4q_s - 2 = 4z_s - 2$. For å sjå dette vil vi fyrst

anta at $q_i = \mathbf{h}_i \cdot \mathbf{z} \neq 0 \; \forall \; |\mathbf{h}_i|_1 \leq 4$. Kravet vert då at $cN \neq q_i, \; \forall \; c \in \mathbb{Z}$, og for å finne ein N som garantert oppfyller dette kravet treng vi berre å velgje ein N som ligg mellom dei to største elementa i $\mathbf{q}$, og der $N > \max(q_i)/2$. Dersom vi let alle elementa i $\mathbf{q}$ vere sortert slik at $q_i \leq q_{i-1}$ får vi at det største elementet er $q_m = 4z_s$ og det nest største elementet er $q_{m-1} = 3z_s + z_{s-1}$. Differansen $q_m - q_{m-1}$ blir

$$4z_s - 3z_s - z_{s-1} = z_s - z_{s-1} = \epsilon_{s,s-1} \geq 3.$$

Ut i frå dette ser vi at om vi vil garantere $q_{m-1} < N < q_m$ kan vi velgje

$$N = q_m - 2 = 4z_s - 2. \tag{25}$$

Eg nemnde over at algoritmen er tilpassa konstruksjonane r1, men om vi brukar konstruksjonane b1, r2 eller b2 vil vi ha nokre endringar. Fyrst og fremst har vi for b1 og b2 at, sidan dei genererer golomblinjalar modulo $p^2 - 1$, får vi i linje 4 $b = p^2 - 1$. Vidare, for r1 og b1 er det, ut i frå Teorem 2.3, naturkeg å velgje ei nedre grense for $N \geq q_s + 2$. For r2 og b2 vil nedre grense for N bli endra til $N \geq z_1 + 1$. Dette ser vi ut i frå grunngjevinga over for å bruke r2 og b2, som nettopp er fordi vi her vil kunne finne ein $N = z_i \pm 1$.

Dersom vi har køyrt eit søk for ein av konstruksjonane har vi funne ein N_{god}. Denne kan vi vidare bruke som ei øvre grense for vidare søk med dei tre andre metodane for å kunne innskrenke søkeområdet. Eg har i denne oppgåva brukt metode r1 som fyrste søk.

Ein annan måte å avgrense søket på spring ut i frå at dei golombkonstruksjonane vi har brukt dannar golomblinjalar for kvart primtal p, og med tal på merker avhengig av dette. Vi vil no sjå på tilfellet for konstruksjonen r1, men argumentasjonen vil vere tilsvarande for dei tre andre konstruksjonane. La p_i og p_{i-1} vere to primtal der

$$p_{i-1} < p_i, p_i - p_{i-1} = \rho$$

og ingen primtal ligg mellom. For p_i og p_{i-1} konstruerer vi to deltasekvensar $\mathbf{a}_i$ og $\mathbf{a}_{i-1}$ med lengde $\sigma_i = p_i - 2$ og $\sigma_{i-1} = p_{i-1} - 1$. Avstanden mellom desse deltasekvensane vil då også vere lik ρ. For å danne ein deltasekvens $\mathbf{a}_i^1$ med $\sigma_i - 1$ element kan vi trunkere $\mathbf{a}_i$ og set

$$\mathbf{a}_i^1 = (a_1, a_2, \ldots, a_{\sigma-1})^T$$

. Denne prosessen kan vi gjenta inntil vi har alle deltasekvensar $\mathbf{a}_i^j; 1 \leq j \leq \rho - 1$.

Ut i frå Teorem 3.2 vil vi, for dei deltasekvensane $\mathbf{a}_j$ konstruert på denne måten, ha at grada $\delta_j \geq \delta_i$. Gitt ein N får vi då at ein $\mathbf{z}_j = (1, \mathbf{a}_i^j)^T$ der $\mathbf{a}_i^j = (a_1, \ldots, a_{\sigma-j})^T$ vil ha same grad eller betre enn ein $\mathbf{z}_i$ generert av $\mathbf{a}_i = (a_1, \ldots, a_\sigma)$. Har vi funne beste N_i for $\mathbf{z}_i$ vil derfor denne N_i kunne brukast som øvre grense i utrekningane for $\mathbf{z}_{i-1}$ og med den N_i.

Ut i frå det same teoremet får vi ei nedre grense for N_i lik N_{i-1}. Dette kan vi også utnytte etter å ha funne ein god N_i. For å finne ei nedre grense for N for alle $\mathbf{z}$ konstruert ut i frå den same deltasekvensen som $\mathbf{z}_i$ vil vi evaluere $\mathbf{z}_{i-\rho+1}$. $N_{i-\rho+1}$ vil no vere ei nedre grense for alle $N_j, i - \rho + 1 < j < i$.

4.5 Kompleksitet

Hovudtyngda av utrekningane ligg i Algoritme 4. Algoritme 1 er også krevjande, men den vert utført kun ein gong for kvar $\mathbf{z}$ testa, og vert dermed utført totalt s gongar. Likevel er det nyttig å sjå på kva som skjer her. Eg har allereie sagt at ein av tinga algoritmen gjer er å finne alle $\mathbf{h}$ der $|\mathbf{h}|_1 < \hat{\delta}$. For å kunne finne kompleksiteten til utrekningane er det derfor essensiellt å vite kor mange ulike $\mathbf{h}$ vi treng å rekne ut. Dette er likt talet på trigonometriske monom av grad mindre enn 5 og vi nyttar eit resultat vist i [18].

Ser vi kun på dei $\mathbf{h}$ der $|\mathbf{h}|_1 = d$ får vi

$$\tau(s,d) = \sum_{i=1}^{\min s,d} \left(\begin{array}{c} s \\ i \end{array} \right) \left(\begin{array}{c} d-1 \\ i-1 \end{array} \right) 2^i$$

ulike $\mathbf{h}$. Vi har her brukt konvensjonane

$$\left(\begin{array}{c} a \\ b \end{array} \right) = 0 \text{ for } b > a$$

og

$$\left(\begin{array}{c} a \\ 0 \end{array} \right) = 1.$$

Vi definerer også $\tau(s,0) = 1$. Vi vil her i frå også rekne med at $s > d$ og vi er i tillegg kun interesserte i $d \leq \delta - 1 = 4$. Om vi ser på alle $\mathbf{h}$ der $|\mathbf{h}|_1 < \hat{\delta}$ får vi då det totale talet på ulike $\mathbf{h}$, inkludert $\mathbf{h} = \mathbf{0}$, lik

$$t(s,\hat{\delta}) = \sum_{j=0}^{\hat{\delta}} \tau(s,j)$$

$$= \sum_{j=0}^{\hat{\delta}-1} \sum_{i=0}^{\hat{\delta}-1} 2^i \left(\begin{array}{c} s \\ i \end{array} \right) \left(\begin{array}{c} j-1 \\ i-1 \end{array} \right)$$

$$= \sum_{i=0}^{\hat{\delta}-1} 2^i \left(\begin{array}{c} s \\ i \end{array} \right) \sum_{j=0}^{\hat{\delta}-1} \left(\begin{array}{c} k-1 \\ i-1 \end{array} \right)$$

Ved å nytte Pascals regel på siste summasjonen får vi at

$$t(s,\hat{\delta}) = \sum_{i=0}^{\hat{\delta}-1} 2^i \left(\begin{array}{c} s \\ i \end{array} \right) \left(\begin{array}{c} \hat{\delta}-1 \\ i \end{array} \right). \tag{26}$$

I våre utrekningar ser vi bort i frå to typar $\mathbf{h}$ vektorar. Dersom $\mathbf{h} = \mathbf{0}$ får vi triviell løysing for (12), og dette er alltid tilfredsstillt. Alle vektorar av typen $\mathbf{h}' = -\mathbf{h}$ der $\{\mathbf{h}', \mathbf{h}\} \in \Lambda^{\perp}$ fordi dei vil gi identiske resultat for (15). På grunn av symmetri i latticet vil vi for ein

kvar $\mathbf{h} \neq \mathbf{0}$ ha ein $\mathbf{h}' = -\mathbf{h}$ og vi kan derfor sjå bort i frå halvparten. Vi vil derfor måtte ta med maksimalt

$$m = \frac{t(s, \hat{\delta} - 1) - 1}{2} = \sum_{i=1}^{\hat{\delta}-1} 2^{i-1} \begin{pmatrix} s \\ i \end{pmatrix} \begin{pmatrix} \hat{\delta} - 1 \\ i \end{pmatrix} \tag{27}$$

ulike $\mathbf{h}$. For ein generell $\hat{\delta}$ vil vi få $m = O(s^{\hat{\delta}-1})$, som gjer at vi for $\hat{\delta} = 5$ får $m = O(s^4)$, og kvar evaluering der $\mathbf{q}$ i Algoritme 4 inngår kan derfor krevje $O(s^4)$ operasjonar.

Dette uttrykket er essensielt, både for tidsbruken til programmet, og for krav til lagringsplass. Eg vil fyrst sjå på minnebruken til programmet. I algoritmen vert det, som nemnd, lagra to store vektorar $\mathbf{q}$ og $\Delta\mathbf{q}$, der begge desse vil vere av lengde m. På grunn av at desse vert overskrivne i prosessen for kvar ny $\mathbf{z}$ vektor vert den totale lagringsplassen $2m$. For dimensjonar $s \leq 100$ vil dette vere godt innanfor det moderne datamaskiner kan takle. Dersom vi derimot skulle lagre alle $\mathbf{h}$ i ei matrise H, som skildra i Algoritme 1, vil lagringsplassen bli $O(s^5)$.

Viktigare vert det då at ein kan redusere tidsbruken. Her også er uttrykket (26) viktig, i og med at $(\mathbf{h} \cdot \mathbf{z})$mod N vert testa m gongar for kvar N. Sidan utrekningane vert utført som $\mathbf{q}$mod N treng vi høgst m operasjonar for kvar N Ved, som skildra i Algoritme 1, å lagre vektorane $\mathbf{q}$ og $\Delta\mathbf{q}$ treng vi kun å rekne ut $\mathbf{q} + \Delta\mathbf{q}$ for å oppdatere $\mathbf{q}$. Dette tek m operasjonar for kvar t i algoritme 4, men sidan dette ikkje inngår i den indre loopen har dette mindre å seie for tidsbruken.

Vidare er det nødvendig å vite kor mange N som totalt sett vert evaluert. I verst mogleg tilfelle har vi at beste $N = N_{max}$. Frå Teorem 3.3 har vi at ein golomblinjal $\mathbf{a} \in GL_3$ av lengde σ garantert er deltasekvens av grad δ dersom $2a_1 - a_\sigma > \delta - 1$. Ein $\mathbf{z}$ konstruert frå ein slik deltasekvens vil då, for konstruksjon r1 og b1, ha dimensjon $s = \sigma + 1$ og for konstruksjon r2 og b2 vil den ha dimensjon $s = \sigma$. Dersom $\mathbf{a}$ er danna ved å forskuve ein $\mathbf{a}'$,der $a'_1 = 0$, med ein konstant distanse c, får vi

$$2(a'_1 + c) - (a'_\sigma + c) \geq 4$$

$$2c - a'_\sigma - c \geq 4$$

$$c \geq a'_\sigma + 4.$$

Sidan $a'_1 = 0$ vil vi her få $a_1 = c = a'_\sigma + 4$ og vi får

$$a_\sigma < 2a'_\sigma + 4. \tag{28}$$

I tillegg har vi også at golomblinjalen $\mathbf{a}'$ har ulik lengde for ulik konstruksjon gitt eit primtal p. For Ruzsa konstruksjonen får vi ein golomblinjal med $p - 1$ merker og lengde $G(p-1) \leq p(p-1)$. For å få $\mathbf{a}$ med minste avstand 3 må vi fjerne høgst to merker, og vi får då i verste fall at $\sigma = p - 3$, noko som gir lengde $a'_s \leq G(\sigma) = (\sigma + 3)(\sigma + 2)$. Tilsvarande argument for Bose konstruksjon, som dannar ein golomblinjal med p merker og lengde $p^2 - 1$, får vi at $\mathbf{a}'$ har lengde $a'_s \leq \sigma^2 - 4\sigma - 1$.

Vidare vil eg kun sjå på konstruksjonsmetoden r1, i og med at eg har brukt denne som grunnlag for fyrste søk. For dei tre andre konstruksjonane har eg brukt resultata frå

denne som øvre grense, og dei vil derfor normalt ikkje fylgje det mønsteret vi ser her. Vi har som kjend for r1 at $s = \sigma + 1$. På grunn av testen i linje 9 i Algoritme 4 får vi at $\mathbf{z} = (q_1, \ldots, q_s)^t$ garantert dannar ein latticeregel av grad $\delta \geq 5$ for ein eller annan N. Dette gjeld i alle fall når

$$\begin{aligned} z_s &= 2a'_\sigma + 4 \\ &= 2(s+2)(s+1) + 4 \\ z_s &= 2s^2 + 6s + 8. \end{aligned}$$

Frå (25) får vi at $N_{max} = 4z_s - 2$, og for r1 har vi $N_{min}(z_s) = z_s + 2$. For dette verste fall tilfellet får vi lågaste N_{min} for $z_s = 2s^2 + 6s + 8$, som gir

$$N_{min} = 2s^2 + 6s + 10.$$

Algoritmen går inntil $N_{min}(z_s) = N_{max}$, og vi let z_s gå frå $z_s = 2s^2 + 6s + 8$ til $z_s = N_{max} - 2$. Vi har då at

$$\begin{aligned} N_{max} &= 4z_s - 2 \\ &= 8s^2 + 24s + 30 \end{aligned}$$

Dersom beste $N = N_{max}$ gir dette oss eit uttrykk for verst mogleg tal på N vi må evaluere gitt dimensjon s og konstruksjon r1.

$$\begin{aligned} &\frac{(N_{max} - N_{min})(N_{max} - 2 - (2s^2 + 6s + 8))}{2} \\ &= \frac{36s^4 + 216s^3 + 564s^2 + 720s + 400}{2} = O(s^4) \end{aligned}$$

.

Vi vil få eit tilsvarande tal for konstruksjonane r2, b1, og b2. Sidan vi for kvar N testar $\mathbf{q} (\text{mod } N)$ får vi med dette $O(s^4) \cdot O(s^4) = O(s^8)$ operasjonar. Til slutt observerer vi at vi i og med at vi har brukt konstruksjonar som gir sykliske golomblinjalar, og derfor av orden s ulike golomblinjalar for kvar konstruksjon. Dette gir oss at vi i verste fall vil måtte utføre totalt $O(s^9)$ operasjonar.

Dette er eit 'verst mogleg' estimat og baserer seg på at vi finn den beste N som den siste vi undersøkjer. I tillegg må vi også ha at for alle N vi evaluerer må det vere den siste q_i frå Algoritme 4 som gjer at testen feilar. Vi vil derfor kunne forvente ei faktisk køyretid som er lågare enn dette.

Det er likevel betre enn det vi ville forvente ved ikkje å nytte metoden skildra i Algoritme 1. Om vi skulle kalkulere (15) direkte for kvar N og alle $\mathbf{h}$ ville vi fått ei køyretid på orden $O(s^10)$ i staden for $O(s^9)$.

5 Eksperimentelle resultat

5.1 Resultat

For latticereglar av dimensjon $s \leq 10$ for utvida grad $\delta = 5$ har dei optimale latticereglane av rank 1 allereie blitt utrekna [29], og vi kan bruke desse som samanlikningsgrunnlag. Søket for $s = 10$ tok ca 26.5 timar, noko som også tydeleg viser at å gå ut over $s = 10$ med fullstendig søk ikkje er vegen å gå. Som vi ser av tabellen under har vi for $s = 10$ ei total køyretid på 19.35 sekund, som er langt betre enn for eit fullstendig søk. For dimensjon $s > 10$ kan vi bruke eit resultat gitt av Cools og Sloan [6], som seier at ei asymptotisk nedre grense for N er $N \geq N_{ME} = 2(s + 1/2)^2 + 1/2$ for grad $\delta = 5$.

CPU tida oppgitt her er den samla tida brukt på å evaluere $\mathbf{z}$ danna ved alle dei ulike konstruksjonane eg har nytta og gir derfor eit bilete av kor godt dei heuristiske metodane brukt i dette programmet fungerer. Programmeringsspråket eg har brukt i mitt program er Java, kompilert med Eclipse Compiler 0.A48, og programmet vart køyrt på ei datamaskin med 32 kjerner på 2.00 GHz kvar. Programmet er parallelisert slik at kvar genererte deltasekvens går på ein tråd, som gir oss at for deltasekvensar generert frå ein σ dimensjonal golomblinjal $\mathbf{a} \in GL_3$, vil vi få σ trådar. Køyretida som her er oppgitt er samla CPU-tid for alle desse trådane, og den reelle køyretida var derfor ein del kortare.

Alle $\mathbf{z}$ er normalisert slik at dei har 1 som fyrste element. For konstruksjonane r1 og b1 treng vi ikkje gjere noko i og med at alle $\mathbf{z} = (1, \mathbf{a})^T$ i utgangspunktet. For r2 og b2 har vi brukt operasjonen (24).

Tabell 1: Tabell over latticereglar

s	N	N_{ME}	N_{opt}	Konstr.	$\mathbf{z}$	CPU-tid
1	5	5		triviell	(1)	0.11s
2	13	13		triviell	(1,5)	0.15s
3	27	25	27	alle	(1,5,8)	0.10s
4	46	41	45	r1	(1,6,16,19)	0.80s
5	69	61	69	b2	(1,4,13,19,29)	1.69s
6	109	85	103	r1	(1,7,12,16,38,41)	2.46s
7	163	113	130	b1	(1,15,20,38,42,48,51)	4.96s
8	244	145	168	r1	(1,21,40,72,85,89,97,103)	6.88s
9	298	181	209	b2	(1,13,34,57,76,79,107,117,122)	12.41s
10	380	221	268	r2	(1,24,66,69,82,96,104,115,143,158)	19.35s
11	495	265		b2	(1,20,52,63,121,170,176,180,188,217,230)	49.64s
12	593	313		b2	(1,42,78,82,145,151,154,204,211,241,259, 272)	72.98s
13	707	365		b2	(1,9,26,32,81,97,111,154,165,202,205,241, 279)	129.30s
14	829	421		b2	(1,130,144,208,216,241,246,264,293,308, 315,334,354,399)	199.37s
15	979	481		b2	(1,6,98,114,125,194,231,246,278,282,295, 336,369,436,480)	510.60s
16	1137	545		b2	(1,19,34,58,225,237,253,285,366,370,407, 412,457,468,554,562)	647.82s
17	1278	613		b2	(1,72,101,120,151,244,284,306,310,441,490, 495,523,580,601,607,618)	1277.47s
18	1485	685		b2	(1,142,201,250,267,281,289,331,396,405, 416,437,528,588,604,628,699,724)	2167.48s
19	1655	761		b2	(1,38,137,156,160,214,250,258,322,362,377, 403,451,512,529,595,600,622,799)	1888.14s
20	1805	841		r2	(1,60,70,86,161,169,199,249,354,385,407, 422,426,469,688,746,785,819,842,863)	1924.09s
21	2009	925		b2	(1,107,163,209,235,267,277,298,336,380,393, 409,416,428,434,611,620,631,740,816,883)	6651.03s

Tabell over latticereglar [forts.]

s	N	N_{ME}	Konstr.	**z**	CPU-tid
22	2309	1013	b1	(1,193,239,277,354,379,390,412,418,422,430, 439,453,483,509,512,528,578,620,635,640,687)	9738.25s
23	2618	1105	r2	(1,14,132,166,199,225,295,341,446,485,510,640, 697,728,746,818,822,845,1074,1080,1135,1165, 1215)	7761.13s
24	2813	1201	r2	(1,38,55,133,160,169,235,242,286,299,443,505,524, 672,828,990,1021,1031,1051,1073,1079,1167,1171, 1384)	5144.26s
25	2917	1301	r2	(1,27,42,71,78,103,152,165,185,274,313,324,380, 600,646,712,931,990,1004,1121,1131,1372,1375, 1410,1427)	4859.82s
26	3137	1405	r2	(1,10,57,71,163,339,426,441,448,469,524,634,658, 785,812,876,930,935,1066,1112,1165,1178,1190, 1369,1411,1495)	6228.27s
27	3409	1513	b2	(1,46,134,214,245,255,289,317,406,506,538,565, 679,703,763,772,805,941,1091,1128,1147,1473, 1538,1541,1589,1683,1701)	26787.45s
28	3613	1625	r2	(1,9,52,97,173,266,296,313,447,754,790,804,876, 891,1093,1121,1159,1277,1311,1344,1357,1482, 1487,1613,1708,1728,1772,1793)	32185.50s
29	3983	1741	b2	(1,20,34,89,186,234,270,294,431,462,567,633,674, 749,788,953,1068,1206,1258,1379,1438,1483,1595, 1602,1645,1730,1745,1904,1986)	75760.5s
30	4416	1861	r2	(1,61,88,386,399,533,539,666,670,699,739,749, 841,857,885,894,951,1229,1322,1353,1474,1520, 1756,1842,1868,1940,2030,2045,2065,2110)	81553.98s
31	4736	1985	b2	(1,57,230,280,415,446,455,491,552,674,744,747, 795,818,855,964,1106,1126,1303,1320,1402,1483, 1496,1525,1555,1588,1704,1789,1824,2110,2193)	50264.89s
32	4937	2113	r2	(1,34,72,95,271,392,448,493,551,805,817,832,849, 858,937,948,958,1024,1164,1194,1387,1411,1439, 1572,1597,1922,2032,2095,2181,2212,2230,2384)	50051.68s
33	5103	2245	r2	(1,61,193,212,305,329,371,379,444,454,614,631, 651,720,930,934,1060,1523,1530,1556,1698,1704, 1729,1747,1776,1785,1931,2110,2132,2218,2286, 2433,2509)	32322.59s
34	5498	2381	b2	(1,24,45,111,197,255,388,445,549,668,757,855,895, 1130,1160,1166,1230,1297,1309,1406,1626,1645, 1784,1862,1983,2003,2145,2233,2292,2388,2613, 2676,2690,2728)	70661.62s

Tabell over latticereglar [forts.]

s	N	N_{ME}	Konstr.	**z**	CPU-tid
35	5729	2521	b2	(1,92,133,221,273,283,346,373,386,391,447,522, 558,634,642,677,688,702,884,994,1122,1202,1392, 1412,1742,1819,1847,1863,1946,2052,2396,2428, 2628,2806,2837)	251346.54s
36	6534	2665	r2	(1,87,152,203,269,277,316,337,360,556,765,797, 837,944,972,1050,1083,1205,1448,1467,1575,1756, 1802,1898,2066,2222,2227,2481,2602,2764,2944, 3046,3056,3076,3131,3145)	318454.44s
37	6923	2813	r2	(1,21,146,180,240,491,725,763,977,1239,1247, 1262,1579,1656,1660,1689,1734,1791,1949,2022, 2057,2097,2113,2125,2249,2295,2409,2613,2668, 2802,2863,2942,3237,3327,3403,3440,3453)	240606.16s
38	7132	2965	b2	(1,264,281,476,540,613,653,664,1178,1241,1319, 1368,1438,1519,1529,1749,1761,1776,1808,1968, 2063,2113,2222,2517,2734,2748,2786,2809,2972, 3054,3060,3154,3185,3230,3455,3476,3483,3520)	176485.32s
39	7542	3121	b2	(1,453,462,592,610,723,780,994,1016,1078,1126, 1170,1222,1308,1471,1545,1744,1916,2020,2028, 2308,2409,2429,2535,2551,2570,2634,2649,2675, 2688,2867,2923,3277,3338,3345,3355,3511,3620, 3707)	576120.16s
40	7934	3281	b2	(1,121,155,283,289,446,493,500,638,786,878,966, 1220,1251,1268,1284,1328,1346,1386,1477,1497, 1789,1803,1813,2047,2083,2229,2362,2719,2896, 2908,2938,3124,3284,3345,3377,3536,3608,3677, 3855)	540520.10s
41	8232	3445	b2	(1,29,128,283,428,540,629,649,664,835,1067,1310, 1353,1364,1390,1424,1614,1714,1985,2078,2204, 2385,2399,2404,2510,2567,2629,2747,2878,2899, 3117,3186,3192,3253,3358,3385,3533,3572,3636, 3654,4003)	770956.41s
42	9092	3613	r2	(1,43,56,373,384,560,773,978,1102,1150,1210, 1320,1426,1477,1681,1734,2201,2291,2475,2557, 2697,2706,2724,2877,2950,3169,3186,3240,3315, 3320,3403,3545,3597,3751,3884,4036,4098,4205, 4246,4309,4348,4486)	1170769.50s
43	9457	3785	b2	(1,264,271,293,307,362,590,594,1199,1327,1438, 1584,1631,1740,1822,2008,2137,2160,2374,2391, 2620,2822,2827,2990,3030,3039,3399,3472,3550, 3626,3902,3939,3950,4124,4224,4294,4304,4345, 4391,4449,4465,4483,4641)	630724.93s

Tabell over latticereglar [forts.]

s	N	N_{ME}	Konstr.	**z**	CPU-tid
44	9846	3961	r2	(1,154,266,281,301,411,519,605,849,949,1001, 1011,1029,1178,1199,1378,1494,1570,1582,1672, 1676,1753,1812,1955,1985,2024,2038,2596,2636, 2691,3321,3389,3438,3462,3677,3924,3950,4256, 4292,4303,4421,4643,4756,4762)	294043.02s
45	10169	4141	b2	(1,51,211,333,358,693,823,833,982,1287,1291, 1476,1519,1547,1613,1682,1718,1837,1868,1921, 2010,2288,2311,2391,2397,2565,2585,2626,2682, 2737,2830,2929,2950,3358,3445,3747,4013,4171, 4254,4526,4672,4716,4742,4779,4790)	1916898.84s
46	10995	4325	b2	(1,87,124,195,250,352,897,1073,1097,1385,1602, 1763,1825,1965,2117,2124,2184,2462,2581,2642, 2885,2895,3132,3202,3229,3329,3618,3722,3890, 4022,4103,4238,4290,4333,4355,4471,4505,4617, 4946,4997,5077,5115,5268,5313,5409,5424)	2261896.71s
47	11581	4513	r2	(1,373,594,640,718,1025,1154,1204,1547,1556, 1599,1603,1711,1737,2135,2172,2202,2272,2489, 2660,2715,2725,2740,2968,2999,3056,3061,3172, 3223,3244,3358,3418,3516,3527,3798,4163,4278, 4752,4878,4946,4949,5158,5244,5261,5363,5570, 5653)	1511214.62s
48	11711	4705	b2	(1,149,258,322,327,439,442,656,802,1039,1056, 1078,1614,1826,1980,2015,2081,2142,2255,2620, 2724,2853,3009,3069,3076,3102,3303,3528,3735, 3925,4055,4089,4151,4199,4207,4302,4470,4609, 4692,4720,4747,5270,5452,5470,5611,5617,5754, 5832)	829771.10s
49	11952	4901	b2	(1,94,131,214,264,268,280,744,753,867,1030, 1051,1418,1461,2091,2164,2190,2267,2524,2727, 3168,3313,3549,3618,3690,3742,3868,3901,4003, 4111,4195,4382,4427,4543,4546,4728,4745,4833, 4973,4998,5194,5337,5351,5416,5438,5662,5694, 5917,5935)	1165512.31
50	12446	5101	b2	(1,131,193,243,400,514,653,782,857,898,959,997, 1339,1365,1523,1577,1814,1970,2159,2246,2320, 2399,2472,2518,2549,2659,2832,3139,3150,3164, 3208,3438,3966,4058,4594,4601,4739,4773,4857, 4942,5161,5169,5294,5347,5613,5898,6004,6044, 6072,6220)	1313081.78s

5.2 Diskusjon

Eit naturleg mål for kvaliteten til latticereglane konstruert med vår metode vil vere å samanlikne N mot den nedre grensa gitt av Cools og Sloan [6] på $N \geq N_{ME} = 2(s+1/2)^2 + 1/2$ for grad $\delta = 5$, og der dei er kjende, også opp mot N_{opt} som er verdien for optimal latticeregel av gitt dimensjon s. Dersom $N(s)$ er av same orden som $N_{ME}(s)$ vil $\sqrt{N}$ vere lineær. Som ei samanlikning vil eg også plotte $\sqrt{N_{ME}}$.

I tillegg kan det vere interessant å samanlikne med ei øvre grense for N avhengig av dimensjon s. Generelt har vi, som vi vil sjå under, at deltasekvensane basert på Bose konstruksjonen dannar dei beste latticereglane i denne oppgåva, og eg vil derfor samanlikne med ei øvre grense basert på dette. Å finne ein god $N_{max}(s)$ for b2 er ikkje heilt triviellt, men på grunn av at denne metoden som nemnd i førre kapittel er analog med å trunkere bort z_1 frå ein $\mathbf{z}$ generert ved b1 kan vi bruke øvre grense for denne konstruksjonen.

Her vil eg fylgje same utrekninga som i kompleksitetsanalysa i føregåande kapittel. Frå likning (25) har vi at $N \leq 4z_s - 2$. Bose konstruksjonen dannar golomblinjalar $\mathbf{a}'$ med p merker og med alle merker $a'_i < p^2 - 1$. Dersom vi må fjerne to merker for at vi skal ha $\epsilon_i, j \leq 3$ får vi at talet på merker for den resulterande golomblinjalen blir $\sigma = p - 2$, og for b1 har vi $\mathbf{z} = (1, \mathbf{a})^T$ som gir $s = \sigma + 1 = p - 1$. Vi nyttar så likning (28) og vi får at

$$z_s = a_\sigma < 2(p^2 - 1) + 4$$

$$z_s < 2s^2 + 4s + 4$$

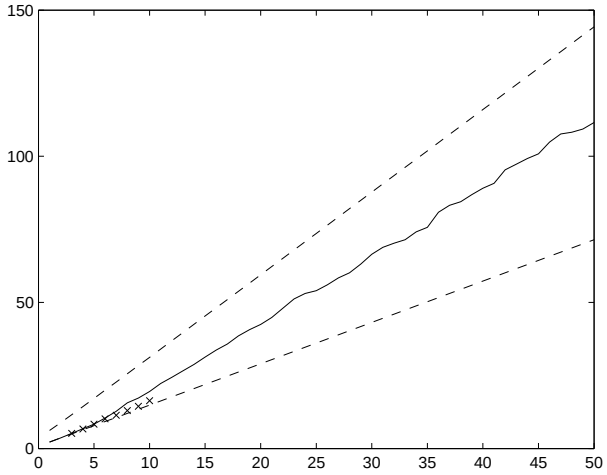
Dette gir oss ei øvre grense $N_{max}(s) < 8s^2 + 16s + 14$. På grunn av at denne øvre grensa er av orden $O(s^2)$ må også N vere av orden $O(s^2)$, men det som er interessant er å sjå om resultata vi har er nærmare øvre enn nedre grense.

I grafen under er $\sqrt{(N)}$ representert ved den heiltrekte linja, høvesvist $\sqrt{N_{max}}$ og $\sqrt{N_{ME}}$ med dei to stripla linjene øverst og nederst, og punkta markert med 'x' er dei optimale latticereglane for $s \in 3, 10$.

Vi legg fyrst merke til at $\sqrt{N}$ er nokså nært ein lineær funksjon, men med relativt stor stigning i høve til $\sqrt{N_{ME}}$. Vi ser med det at $O(s^2)$ er korrekt, men vi har $N \approx 2N_{ME} \approx 5s^2$, og ein faktor på 2 er relativt høgt. Samanliknar vi så med dei kjende optimale N_{opt} støttar desse resultata noko opp om dette. Vi ser en svak tendens til at dei optimale latticereglane ligg nærmare inn til grensa N_{ME} dei gjer til våre N, men med lite materiale å gå etter kan vi ikkje konkludere med noko sikkert. Det gir oss likevel motivasjon for å ville undersøke andre konstruksjonar av $\mathbf{z}$ for å om mogleg minske dette gapet.

Vi samanliknar så i tillegg med den øvre grensa skildra ovanfor, og ser at den verkelege verdien av $\sqrt{N}$ ligg noko nærmare $\sqrt{N_{max}}$ enn $\sqrt{N_{ME}}$. Håpet var at vi skulle finne at N ligg nærmare nedre grense, og ut i frå dette har vi framleis ein del å gå på. Samanlikninga med dei resultata vi har sett for N_{opt}, som ligg nærmare N_{ME} enn mine N, bekreftar dei også dette.

For å få eit bilete av kor langt ein kan nå med god reknekraft og tid vil vi no sjå på den faktiske køyretida til programmet. Frå førre kapittel har vi eit 'dårlegast mogleg'



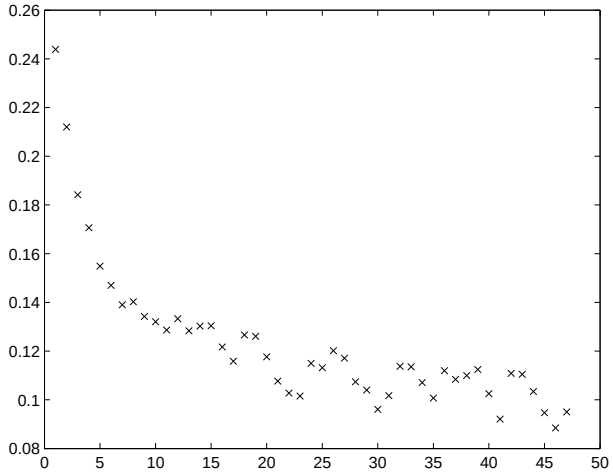
Figur 1: Grenser for N

estimat som seier at køyretida vil ligge på $t = O(s^9)$ eller lågare. Under antakinga at dette attspeglar den faktiske køyretida vil vi plotte $t^{1/9}/s$. Dersom vi då observerer ei stabilisering om ein konstant verdi kan vi konkludere med at estimatet er korrekt.

Ut i frå den fyrste grafen her ser vi at dette estimatet er alt for høgt, noko som vi allereie har forutsett. Det finnast då også fleire grunnar til at køyretida skulle vere lågare enn estimatet på $O(s^9)$. For det fyrste kan vi la Algoritme 4 vere implementert slik at den avbryt evalueringa av gitt N når den finn ein $q_i (\text{mod } N) = 0$, noko som gjer at vi sjeldant får ei full køyring av den indre loopen i algoritmen. Som oftast får vi at den indre loopen køyrer fullstendig kun dersom vi finn ein god N. For alle andre tilfelle vil ein finne ein eller annan q_i som gir $q_i \equiv 0 (\text{mod N})$, og programmet avbryt køyringa.

I tillegg vil vi som oftast kunne finne ein god N for gitt $\mathbf{z}$ lenge før vi har nådd $N = N_{max}$. Med ei meir dynamisk grensesetjing, som vist i Algoritme 4, der vi let den nye N_{god} bli brukt som øvre grense for resten av køyringa. Når vi i tillegg brukar denne N_{god} som øvre grense for vidare søk med andre konstruksjonar vil vi forvente å sjå ein markant reduksjon i køyretida i høve til dette estimatet.

Dersom vi ser på dei resultata vi har viser det seg at den eigentlege køyretida er nærmare $t = O(s^7)$. Dette ser vi ut i frå grafen under, der vi har plotta $g(s) = t^{1/7}/s$.



Figur 2: Test køyretid for $O(s^9)$

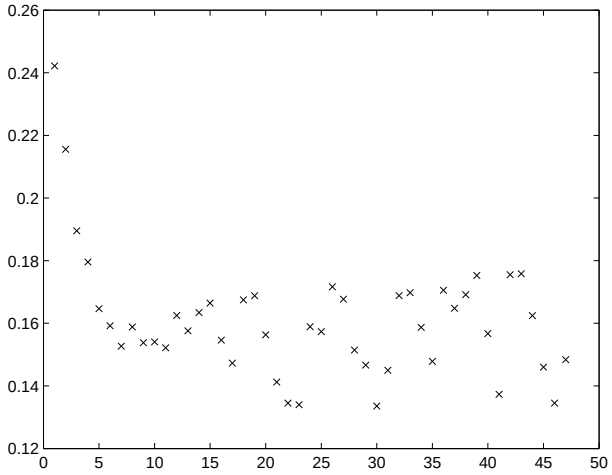
Tabell 2: Fordeling av gode latticereglar

Konstruksjon	Latticereglar
r1	3
r2	15
b1	2
b2	27

På begge desse grafane kan vi sjå ei relativt stor spreiing av køyretida, og det kan sjå ut til å vere oscillerande. Dette er ikkje uventa ut i frå måten $\mathbf{z}$ er blitt konstruert. Som skildra for Algoritme 2 og 3 har vi kun golomblinjalar for kvar p primtal. Vi har kunna utnytte dette slik det er skildra i Kapittel 4.4. For dei $\mathbf{z}$ vektorane generert ut i frå trunkerte deltasekvensar har vi kunne brukt tidlegare funne N_{god} som grenser. Derimot, for dei deltasekvensane konstruert frå utrunkerte deltasekvensar, kan vi ikkje utnytte ein N_{god} funne for ein tilsvarande lenger $\mathbf{z}$ som grense. Dette gjer at køyretida vert tilsvarande lenger for sistnemnde tilfelle, og vi vil derfor også få toppar i køyretida for kvart primtal.

Ut over dette er det også verdt å merke seg kva valet av konstruksjonsmetode har for resultatet. Vi kan setje opp ein tabell, med talet på beste latticereglar funne for kvar konstruksjonsmetode. Eg har ikkje rekna med dei genererte latticereglane for dimensjon $s \leq 3$ i og med at alle konstruksjonane her gav optimale latticereglar.

Det fyrste vi ser ut i frå dette er at dei aller fleste gode latticereglar er konstruert anten med metode r2 eller b2, og dei tilfella der gode latticereglar har blitt funne for



Figur 3: Test køyretid for $O(s^7)$

r1 eller b1 er det for låg dimensjon. Ingen av dei beste latticereglane vi har funne for dimensjon $s > 22$ er generert ved hjelp av r1 eller r2. Vi kan derfor rekne med at dette er eit mønster som også held for $s > 50$, og for vidare utrekningar kan vi sannsynlegvis sjå bort i frå desse konstruksjonane, og med det nesten halvere køyretida.

Når vi så samanliknar balansen mellom latticereglar konstruert av r2 og b2 er det fleire mønster som viser seg. Det fyrste er at det er nesten dobbelt så mange gode latticereglar generert med b2 som med r2. Det kan dessutan sjå ut til at dess høgare dimensjon vi har dess større overvekt er det av konstruksjonar danna ved b2 enn det er av r2. Dette er helder ikkje unaturleg sidan Bose konstruksjonen gir golomblinjalar som er noko kortare enn dei linjalane generert ved Ruzsa konstruksjonen.

Det andre er at dei ikkje er jamt fordelt, men latticereglar konstruert med ein av metodane ligg gjerne i grupper. Ut i frå at vi for dei fleste deltasekvensane må forkorte ein lenger deltasekvens er dette helder ikkje unaturleg. Dersom ein utrunkert deltasekvens $\mathbf{a}$ gir ein god latticeregel $\mathbf{z}$ av dimensjon s er det helder ikkje unaturleg å finne ein god latticeregel ved å trunkere deltasekvensen $\mathbf{a}$. Desse to punkta gir grunnlag for å ville danne meir intelligente metodar for val av rekkjefylgja på evalueringa av dei ulike konstruksjonane, der ein vel å fyrst evaluere den latticeregel som mest sannsynleg gir best N, for så å bruke denne N som øvre grense for vidare evaluering. I tillegg kan vi også vente oss at om vi går ein del høgare opp i dimensjon vil konstruksjonen b2 ta over heilt samanlikna med r2.

6 Konklusjon

I denne oppgåva har vi sett på latticeregla av rank-1 og utvida grad $\delta = 5$, og utvikla algoritmer for å finne gode medlemmer av desse. Til dette har vi brukt ein del tidlegare upublisert teori om deltasekvensar, som bind saman latticeregla av grad $\delta \geq 5$ og teorien om golomblinjalar. På grunn av dette har det vore mogleg å bruke tidlegare utvikla konstruksjonar for golomblinjalar for å generere deltasekvensar av grad 5, og dermed også latticeregla av utvida grad $\delta = 5$. Teknikken med å dele opp utrekninga av grad i to trinn, med utrekninga av $\mathbf{q}$ og $\Delta\mathbf{q}$ i Algoritme 1 og $\mathbf{q} (\mathrm{mod}\ N)$ i Algoritme 4 er nytt i denne oppgåva. Det er i all hovudsak dette som har gjort det mogleg å få algoritmer som er hurtige nok til at vi kunne bruke dei i eit program for søk etter gode latticeregla av høg dimensjon. Desse algoritmane har eg så nytta i eit program for å søke etter rank 1 latticeregla for høg dimensjon.

Dette søket har vore ein suksess ved at vi hurtig har funne relativt gode rank 1 latticeregla opp til dimensjon $s = 50$. At dei er gode ser vi ut i frå at for $s \leq 50$ er alle N av orden $O(s^2)$. Dette er det vi ville forvente ut i frå grensa gitt av Cools og Sloan [6] på $N_{ME} = 2(s + 1/2)^2 + 1/2$, og noko betre enn $O(s^2)$ kan vi derfor ikkje oppnå. Men optimale er dei ikkje, noko vi også har sett ved å samanlikne N med både ei øvre grense N_{max}, nedre grense N_{ME}, og for dimensjon $s \leq 10$ også opp mot optimale N_{opt} funne ved brute force søk.

Vidare undersøkte vi køyretida til dette programmet opp mot ein antatt verste køyretid på $O(s^9)$. Her såg vi at vi kunne forvente ei køyretid på nærmare $O(s^7)$, noko som er signifikant betre enn estimatet for verst mogleg køyretid. I tillegg såg vi at når eit fullstendig søk etter ein optimal latticeregel for dimensjon $s = 10$ tok omtrent 26.5, tok det for mitt program 19.35 sekund på den same datamaskinen å finne ein akseptabel latticeregel av lik dimensjon. Eg vil ut i frå dette derfor føretrekke algoritmen vi har sett i denne oppgåva.

Det søket vi har gjennomført er på ingen måte komplett, og vi har berre fått undersøkt ein liten del av alle moglege deltasekvensar. For det fyrste finnast det fleire måta å konstruere eller finne golomblinjalar, som vidare vil kunne gi oss nye sett av deltasekvensar. For det andre så har vi helder ikkje undersøkt alle moglegheitene for å danne deltasekvensar ut i frå dei spesifike konstruksjonane, men konsentrert oss om å la dei vere så korte som mogleg. Likevel, ut i frå samanlikninga med optimale latticeregla funne for lågare dimensjon kunne vi forvente å finne endå betre konstruksjonar enn dei vi har brukt.

Meir interessant enn å undersøkje fleire latticeregla av dimensjon mindre enn 50, kan det vere å gå endå høgare og finne laticeregla av dimensjon $s > 50$. Men for å klare dette må vi få algoritmene til å gå endå fortare enn dei har gjort til no. Fleire områder i algoritmene kan forbetrast for å gjere de hurtigare. Vi såg ein klar tendens i samanlikninga mellom dei ulike konstruksjonsmetodane for $\mathbf{z}$ at r2 og b2 dannar fleire av dei beste latticeregla funne enn r1 og b2 gjer. Dette motiverer oss til å ville undersøkje om vi kan kutte ut desse konstruksjonane heilt, og dermed redusere mengda av deltasekvensar vi må undersøke. Eit slikt grep vil kunne nesten halvere køyretida. Vi såg også ein slik tendens mellom b2 og r2, der det ser ut til at fleire gode latticeregla er danna med b2 enn

med r2, noko som motiverer oss til å ville bruke b2 som basis for fyrste gjennomkøyring, og bruke resultata frå dette som grense for ei andre gjennomkøyring med r2.

Det vil også vere av interesse å finne latticereglar av høgare trigonometrisk utvida grad enn $\delta = 5$. Dei algoritmene eg har funne for å avgjere om ein latticeregel er av grad $\delta = 5$ kan lett modifiserast til å gjelde $\delta > 5$. På grunn av at (27) gir $m = O(s^{\hat{\delta}-1})$ vil vi dessverre måtte rekne med å sjå ei korresponderande auke i køyretida, og ein vil derfor ynskje å utvikle desse algoritmene vidare for å kunne senke køyretida endå meir. Ein av desse metodane kan vere meir intelligente måtar å utnytte parallelprogrammering enn dei eg har brukt. I tillegg vil arbeid med gensesetjing for N naturleg gjere algoritmene meir effektive.

Eit meir alvorleg problem er at vi no ikkje har nokon algoritmer endå for å automatisk generere deltasekvensar av grad $\delta = 6$. For tilfellet $\delta = 5$ var vi heldige, då det allereie eksisterte gode metodar for konstruksjon av golomblinjalar. Vil vi søke etter deltasekvensar av høgare grad er vi ikkje garantert å vere like heldige.

Kjernen i dette problemet vil vere å finne gode nok krav til ei talrekkje for at den skal vere deltasekvens av grad $\delta = 6$. Dessverre vil ikkje Teorem 3.3 vere tilstrekkjeleg for dette, sjølv om desse krava er nødvendige, og for å kunne gå vidare må ein derfor utarbeide nye krav. Ut i frå Lemma 3.1 er det klart at for ein deltasekvens av grad $\delta = 6$ må vi ha at minste avstand mellom to element må vere $\epsilon \geq 4$. Ut over dette er ikkje nokon andre krav til no kjende.

Oppsumert vil derfor arbeidet vidare i fyrste rekkje ligge i å finne ein analog til Teorem 3.3 for $\delta = 6$. Deretter er det nødvendig å finne konstruksjonar for deltasekvensar av grad $\delta = 6$. På grunn av dimensjonsspøkelset vil det til slutt også vere nødvendig med forbetringar av algoritmen for at søket skal vere effektivt nok.

Referansar

[1] M.D. Atkinson, Santoro N., and Urruita J. Integer sets with distinct sums and differences and carrier frequence assignment for nonlinear repeaters. *IEEE Transactions on communication*, 34:614–7, 1986.

[2] W.C. Babcock. Intermodulation interference in radio systems. *Bell Systems Technical Journal*, pages 63–73, 1953.

[3] R.C. Bose and S. Chowla. Theorems in the additive theory of numbers. *Commentarii Mathematici Helvetici*, 37:141–7, 1962.

[4] H. Conroy. Molecular schrödinger equation viii: A new method for the evaluation of multidimensional integrals. *The Journal of Chemical Physics*, 47:5307–18, 1967.

[5] R. Cools. Program trigdegree. Notat og eit fortran-program for utrekning av trigonometrisk grad.

[6] R. Cools and I.H. Sloan. Minimal cubature formulae of trigonometric degree. *Mathematics of Computation*, 65(216):1583–1600, 1996.

[7] A. Dimitriomanolakis. *Analysis of the Golomb Ruler and the Sidon Set Problems, and Determination of Large, Near-Optimal Golomb Rulers.* PhD thesis, Technical University of Crete, 2002.

[8] P. Erdős and P Turán. On a problem of sidon in additive number theory, and on some related problems. *Journal of the London Mathematical Society*, 16:212–5, 1941.

[9] M. Garry et al. Project ogr, www.distributed.net. Henta 23.mai.2011.

[10] K.K. Frolov. On the connection between quadrature formulas and sublattices of the lattice of integral vectors. *Doklady Akademii Nauk SSSR*, 232:37–41, 1977. (russisk).

[11] E. Hlawka. Zur angenäherten berechnung mehrfacher integrale. *Monatshefte für Mathematik*, 66:140–51, 1962.

[12] N.M. Korobov. The approximate computation of multiple integrals. *Doklady Akademii Nauk SSSR*, 124:1207–10, 1959. (russisk).

[13] A.W. Lam and D.V. Sarwate. On optimum time-hopping patterns. *IEEE Transactions on communications*, 36:380–2, 1988.

[14] B Lindström. An inequality for b2-sequences. *Journal of Combinatorial Theory*, 6:211–2, 1969.

[15] J.N. Lyness. Handskrive notat. Notat med teorem for krav til deltasekvensar av utvida grad 5.

[16] J.N. Lyness. An introduction to lattice rules and their generator matrices. *IMA Journal of Numerical Analysis*, 9:405–19, 1989.

[17] N.C. Metropolis and S.M. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44:335–41, 1949.

[18] I.P. Mysovskikh. On cubature formulas that are exact for trigonometric polynomials. *Doklaty Akademii Nauk SSSR*, 296:28–31, 1987. (russisk).

[19] H. Niederreiter. *Random number generation and quasi Monte Carlo methods*. Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[20] I.Z. Ruzsa. Solving a linear equation in a set of integers i. *Acta Arithmetica*, LXV.3:259–82, 1993.

[21] S. Sidon. Ein satz über trigonometrische polynome und seine anwendungen in der theorie der fourier-reihen. *Mathematische Annalen*, 106:536–9, 1932.

[22] I.H. Sloan. Lattice methods for multiple integration. *Journal of Computational and Applied Mathematics*, 12 og 13:131–43, 1985.

[23] I.H. Sloan and S. Joe. *Lattice methods for multiple integration*. Oxford University Press, 1994.

[24] I.H. Sloan and P.J. Kachoyan. Lattices for multiple integration. In *Mathematical programming and numerical analysis workshop*, number 6 in Proceedings of the Centre for Mathematical Analysis, pages 55–69. Australian National University, Canberra, 1984.

[25] I.H. Sloan and P.J. Kachoyan. Lattice methods for multiple integration: theory, error analysis and examples. *SIAM Journal on Numerical Analysis*, 24:116–28, 1987.

[26] I.H. Sloan and J.N. Lyness. The representation of lattice quadrature rules as multiple sums. *Mathematics of Computation*, 52:81–94, 1989.

[27] I.H. Sloan and L. Walsh. A computer search of rank 2 lattice rules for multidimensional quadrature. *Mathematics of Computation*, 54:281 302, 1990.

[28] T. Sørevik. Notes on delta sequences. Notat med definisjon av deltasekvensar, 2006.

[29] T. Sørevik. Optimale rank-1 latticereglar opp til dimensjon 10. Resultat frå bruteforce søk etter optimale rank-1 latticereglar, 2011.