NTNU
Det skapende universitet

Gröbner-baser og signaturskjemaet Unbalanced Oil and Vinegar

Jarle Kvåle

Master i fysikk og matematikk
Oppgaven levert: Juni 2009
Hovedveileder: Aslak Bakke Buan, MATH
Biveileder(e): Petter Andreas Bergh, MATH

Norges teknisk-naturvitenskapelige universitet
Institutt for matematiske fag

Oppgavetekst

Hensikten med denne masteroppgaven er å studere bruken av Gröbner-baser i kryptografi. Spesielt undersøkes de algebraiske egenskapene til Gröbner-baser og algoritmer for å finne Gröbner-baser. I tillegg tar også oppgaven for seg det multivariate kryptosystemet Unbalanced Oil and Vinegar, introdusert av Goubin, Kipnis og Patarin. Oppgaven undersøker hvordan dette kan angripes, blant annet ved hjelp av Gröbner-baser.

Oppgaven gitt: 26. januar 2009
Hovedveileder: Aslak Bakke Buan, MATH

Sammendrag

Vi har i denne masteroppgaven sett nærmere på Gröbner-baser og signaturskjemaet Unbalanced Oil and Vinegar.

Vi har sett nærmere på Gröbner-basenes definisjoner og sett hvordan Gröbner-baser kan genereres ved Buchbergers algoritme. Videre har vi sett hvordan Gröbner-baser hjelper for å gi resten i divisjonsalgoritmen i den multivariable polynomringen unikhet og indirekte dermed løse idealmedlemskapsproblemet.

Videre undersøkte vi mulighetene for å bruke Gröbner-baser til å lage et offentlig nøkkel-kryptosystem. Foreløpig er det ingen som har greid å lage et slikt kryptosystem som har stått imot visse angrep.

Den neste delen av oppgaven tok for seg signaturskjemaet Unbalanced Oil and Vinegar. Vi har presentert teorien bak UOV og sett nærmere på et enkelt eksempel. I tillegg har vi undersøkt tre angrep på UOV, der det ene var basert på Gröbner-baser. Det viser seg at UOV virker resistent mot disse angrepene gitt at vi tar visse forholdsregler med parametrene. For angrepet basert på Gröbner-baser må systemet være av en viss størrelse for at sikkerheten skal ivaretas.

Avslutningsvis har vi sett på forbedrede algoritmer for å beregne Gröbner-baser. Den første algoritmen vi så på var F_4 -algoritmen som tok i bruk lineæralgebra, mens den andre, F_5-algoritmen, tok utgangspunkt i å kutte ut unødvendige beregninger. Det viser seg at både F_4- og F_5-algoritmen kraftig forbedrer beregningstiden for å finne en Gröbner-basis.

Forord

Denne masteroppgaven er en avslutning på fem fine år på sivilingeniørstudiet for fysikk og matematikk. Det har vært en lærerik opplevelse, både faglig og sosialt. Helt fra første øving i Matematikk 1 til denne avsluttende masteroppgaven har det alltid vært nye og spennende utfordringer. I tillegg har jeg fått mange nye vennskap som jeg håper vil vare livet ut og jeg vil alltid se tilbake på denne tiden som en fantastisk periode i mitt liv.

Jeg vil spesielt takke Jon Inge Kolden for samarbeidet. Det å ha en å diskutere oppgaven med og ikke minst spille bordtennis med i pausene, har gjort dette semesteret mye morsommere. En annen jeg må nevne er veileder Petter Andreas Bergh. Tålmodig veiledning fra start til slutt har gjort denne oppgaven til en fornøyelse å skrive. Til slutt en takk til Aslak Bakke Buan som alltid har vært tilgjengelig.

Innhold

1 Introduksjon **1**
- 1.1 Gröbner-baser . . . 1
- 1.2 Oppbygging av oppgaven . . . 1

2 Multivariate polynomringer **3**
- 2.1 Monom sortering . . . 3
- 2.2 Monome sorteringsmetoder på polynomer . . . 5
- 2.3 Terminologi . . . 5
- 2.4 Divisjonsalgoritme i $k[x_1, \ldots, x_n]$ . . . 6

3 Gröbner-baser **9**
- 3.1 Egenskaper til Gröbner-baser . . . 9
- 3.2 Buchbergers algoritme . . . 11
- 3.3 Bruk av Gröbner-baser . . . 14

4 Problemene med kryptosystem basert på Gröbner-baser **17**

5 Unbalanced Oil and Vinegar **21**
- 5.1 Multivariat kryptografi . . . 21
- 5.2 Generelt om signaturskjema . . . 21
- 5.3 Motivasjon . . . 22
- 5.4 Unbalanced Oil and Vinegar . . . 22
- 5.5 Eksempel på UOV . . . 23
- 5.6 Angrep på UOV . . . 24
- 5.7 Oppsummering . . . 29

6 Forbedringer av Buchbergers algoritme **31**
- 6.1 Faugéres F_4-algoritme for beregning av Gröbner-baser . . . 31
  - 6.1.1 Buchbergers kriterium. Forbedrede F_4-algoritmer . . . 35
  - 6.1.2 Utvelgelsesstrategi . . . 38
  - 6.1.3 Eksempel på f_4-algoritmen . . . 39
  - 6.1.4 Oppsummering av F_4 . . . 40
- 6.2 Faugéres F_5-algoritme . . . 41
  - 6.2.1 Ideen bak F_5-algoritmen . . . 41
  - 6.2.2 Signaturen til et polynom . . . 44
  - 6.2.3 Nytt kriterium . . . 45
  - 6.2.4 Simplifiseringsregler . . . 46
  - 6.2.5 Beskrivelse av algoritmen . . . 47
  - 6.2.6 Eksempel på F_5-algoritmen . . . 51
  - 6.2.7 Oppsummering av F_5-algoritmen . . . 52

7 Oppsummering **53**

1 Introduksjon

1.1 Gröbner-baser

Gröbner-baser er en spesiell type genererende undermengder av et ideal I i en polynomring R. Det viser seg at Gröbner-basene har mange egenskaper som vi ønsker. Blant annet har de egenskaper som gjør at vi kan arbeide i faktorringen $R\backslash I$ like lett som med modulær aritmetikk. Det er en viktig ting i kommutativ algebra at Gröbner-baser alltid vil eksistere og at disse kan beregnes for ethvert ideal gitt en genererende undermengde.

Denne oppgaven ser nærmere på hvordan Gröbner-baser fungerer på et teoretisk nivå, samtidig som vi også prøver å se nærmere på den praktiske bruken av Gröbner-baser. Vi vil også betrakte algoritmer for å beregne Gröbner-baser og hvordan disse kan optimeres til å få en raskere kjøretid.

1.2 Oppbygging av oppgaven

Denne oppgaven begynner med å ta for seg multivariabel algebra og vi ser spesielt på sorteringsmetoder og en divisjonsalgoritme i disse tilfellene. I det neste kapitlet ser vi på hva Gröbner-baser er og hvilke fine egenskaper de bringer med seg. I denne delen tar vi også for oss Buchbergers algoritme, den enkleste måten å beregne en Gröbner-basis.

Deretter ser vi på Gröbner-basers egenskaper til å bygge opp kryptosystemer, og hvilke problemer dette medfører. Vi beveger oss så inn på generell multivariat kryptografi. Nærmere spesifikt kommer vi inn på Unbalanced Oil and Vinegar, et multivariat signaturskjema som ikke tar hensyn til Gröbner-baser i oppbyggingen. Derimot kan Gröbner-baser brukes til å angripe dette systemet, noe vi kommer inn på sammen med to andre typer angrep.

Den siste hoveddelen tar for seg to algoritmer av Faugére som på hver sin måte forbedrer den klassiske algoritmen for å beregne Gröbner-baser. Først ser vi på F_4-algoritmen og hvordan denne bruker lineæralgebra på ulike steg. Til slutt ser vi på F_5-algoritmen og hvordan denne forhindrer unødvendige beregninger.

2 Multivariate polynomringer

En polynomring er en ring som blir formet av mengden polynomer med en eller flere variable med koeffisienter i en ring. Dersom vi ser på enkleste tilfellet først vet vi at et polynom av x med koeffisienter, p_i, i en kropp k er på formen $p = p_m x^m + p_{m-1}x^{m+1} + \cdots + p_1 x + p_o$. Vi sier da at $p \in k[x]$. Vi har da en kommutativ ring $k[x]$ som vi kaller polynomringen i én variabel, x, over k.

Et polynom i n variable $x_1, \ldots, x_n$ med koeffisienter i kroppen k blir definert på samme måte som et polynom med bare en variabel, men notasjonen blir betraktelig mer strevsom. Vi skriver det slik:

$$f = \sum_{\alpha} a_\alpha x^\alpha, \quad a_\alpha \in k,$$

der summen er over et endelig antall n-tupler $\alpha = (\alpha_1, \ldots, \alpha_n)$. Vi har ikke lenger bare potenser av x, men kombinasjoner av x_i-ene. Mengden med alle polynomer i $x_1, \ldots, x_n$ med koeffisienter i k skriver vi $k[x_1, \ldots, x_n]$. Dersom vi bare har et lite antall variable, tar vi oss vanligvis ikke bryet med subskript og skriver for eksempel $k[x, y]$ eller $k[x, y, z]$.

2.1 Monom sortering

For å kunne jobbe effektivt og konsistent med polynomringene er det viktig at det er klargjort hvordan polynomene våre skal sorteres. Som vi vil se senere er det ledende leddet meget viktig i beregningene, og derfor må det klart komme frem hva det ledende leddet faktisk er. Hva skal for eksempel være størst av x_1x_2 og x_2^2? Før vi går nærmere inn på ulike sorteringsmetoder definerer vi først hvilke krav som må tilfredsstilles for at det skal kunne kalles monom sortering:

Definisjon: *En monom sortering $>$ på $k[x_1, \ldots, x_n]$ er en relasjon $>$ på $\mathbb{Z}^n_{\geq 0}$, eller ekvivalent, en relasjon på mengden monomer x^α, $\alpha \in \mathbb{Z}^n_{\geq 0}$, som tilfredsstiller:*

i) *$>$ er en total (eller lineær) sortering på $\mathbb{Z}^n_{\geq 0}$.*

ii) *Dersom $\alpha > \beta$ og $\gamma \in \mathbb{Z}^n_{\geq 0}$, så er $\alpha + \gamma > \beta + \gamma$*

iii) *$>$ er velordnet på $\mathbb{Z}^n_{\geq 0}$. Det betyr at enhver ikke-tom undermengde av $\mathbb{Z}^n_{\geq 0}$ har et minste element under $>$.*

Det finnes mange forskjellige sorteringsmetoder, alle med sine egne fordeler og ulemper. Vi tar for oss de tre viktigste for denne oppgaven.

2.1.1 Leksikografisk sortering (lex)

Definisjon: *La $\alpha = (\alpha_1, \ldots, \alpha_n)$ og $\beta = (\beta_1, \ldots, \beta_n)$ være elementer i $\mathbb{Z}^n_{\geq 0}$. Vi sier at $\alpha >_{lex} \beta$ dersom, i vektordifferansen $\alpha - \beta \in \mathbb{Z}^n$, det første elementet fra venstre ulik null er positivt. Vi skriver $x^\alpha >_{lex} x^\beta$ dersom $\alpha >_{lex} \beta$.*

Eksempelvis ser vi at $xz^2 >_{\text{lex}} y^5z^5$. Vi får nemlig $(1, 0, 2) >_{\text{lex}} (0, 7, 5)$ siden $\alpha - \beta = (1, -7, -3)$. I tillegg ser vi at variablene $x_1, \ldots, x_n$ sorteres vanlig siden $(1, 0, \ldots, 0) >_{\text{lex}} (0, 1, 0, \ldots, 0) >_{\text{lex}} \cdots >_{\text{lex}} (0, \ldots, 0, 1)$.

Dette er den matematiske definisjonen av denne sorteringsmåten, men vi er gjerne vant til å se noe lignende i en ordbok. Setter vi $x_1 = \text{A}$, $x_2 = \text{B}$ og så videre, ser vi at det blir nesten det samme. De eneste forskjellene er at ordene i en ordbok ikke er kommutativ ($\text{ABBA} = x_1 \cdot x_2^2 \cdot x_1 \neq x_1^2x_2^2 = \text{AABB}$), samt at lengre 'ord' blir plassert først i rekken. Et kjapt eksempel er at $x_1^2x_2 > x_1^2$ selv om

'aab' ville kommet bak 'aa' i en ordbok. En annen viktig egenskap er at den høyeste variabelen vil dominere ethvert annet monom som bare har lavere variabler, uavhengig av graden til dette andre monomet. Dersom vi har lex-ordningen $z > y > x$, får vi for eksempel $z >_{\text{lex}} y^8x^5$.

Det er derfor viktig å være oppmerksom på den interne sorteringen til variablene. Det er ikke alltid $x > y > z$. Det er mange forskjellige måter man kan sortere variablene på og gitt en vilkårlig sortering av variablene så finnes det en tilsvarende lex-ordning. Det er $n!$ lex-ordninger av n variable avhengig av hvordan variablene, $x_1, \ldots, x_n$, sorteres.

2.1.2 Leksikografisk totalgradssortering (grlex)

I mange sammenhenger tar vi heller hensyn til den totale graden til monomene og sorterer dem med høyest grad først. En måte å gjøre dette på er leksikografisk totalgradssortering.

Definisjon: *La $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$. Vi sier at $\alpha >_{grlex} \beta$ dersom:*

$$|\alpha| = \sum_{i=1}^{n} \alpha_i > |\beta| = \sum_{i-1}^{n} \beta_i, \text{ eller } |\alpha| = |\beta| \text{ og } \alpha >_{lex} \beta.$$

Vi ser her at grlex sorterer etter total grad. Dersom vi har $x > y > z$ får vi at $xy^3z^2 \sim (1,3,2) >_{\text{grlex}} (2,3,0) \sim x^2y^3$ siden $|(1,3,2)| = 6 > |(2,3,0)| = 5$. Dersom flere ledd har samme grad bruker den lex-ordningen til å skille disse. Eksempelvis får vi at $y^3z \sim (0,3,1) <_{\text{grlex}} (1,3,0) \sim xy^3$ siden $|(0,3,1)| = 4 = |(1,3,0)|$, men xy^3 inneholder variabelen x. Variablene selv, $x_1, \ldots, x_n$, blir sortert etter lex-ordningen da alle disse har samme grad.

2.1.3 Omvendt leksikografisk totalgradssortering (grevlex)

I tillegg til disse to sorteringene har vi en tredje og litt mer uvanlig en, omvendt leksikografisk totalgradssortering. Denne ordningen er litt mindre intuitiv og det tar litt tid å bli vant til den. Grunnen til at ser nærmere på denne er at det for enkelte operasjoner viser seg at dette er den mest effektive sorteringsmetoden. Blant annet er den meget relevant i denne oppgaven da det viser seg at den produserer relativt små Gröbner-baser.

Definisjon: *La $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$. Vi sier at $\alpha >_{grevlex} \beta$ dersom*

$$|\alpha| = \sum_{i=1}^{n} \alpha_i > \sum_{i=1}^{n} \beta_i = |\beta|, \text{ eller } |\alpha| = |\beta| \quad \text{og} \quad \text{den første}$$
$$\text{verdien ulik null fra høyre i } \alpha - \beta \text{ i } \mathbb{Z}^n \text{ er negativ.}$$

På samme måte som grlex sorterer grevlex etter total grad, men for ledd med lik grad blir det annerledes. For eksempel er $x^4y^7z \sim (4,7,1) >_{\text{grevlex}} (4,2,3) \sim x^4y^2z^3$ siden $|(4,7,1)| = 12 > |(4,2,3)| = 9$, på samme måte som grlex. Derimot får vi at $xy^5z^2 \sim (1,5,2) >_{\text{grevlex}} (4,1,3) \sim x^4yz^3$ siden $|(1,5,2)| = |(4,1,3)|$, $(1,5,2) - (4,1,3) = (-3,4,-1)$ og vi ser at den første verdien fra høyre er negativ. Vi legger også merke til at variablene, $x_1, \ldots x_n$, her også vil bli sortert på samme måte som før.

For å forklare forskjellen mellom grlex og grevlex, så ser vi at det er måten man sorterer ledd med lik grad. Når dette skjer vil grlex gå etter den største variabelen og sortere etter den største graden på denne. Dette i motsetning til grevlex, som går etter den minste variabelen og sorterer

etter den minste graden på denne. For eksempel får vi at $x^5yz >_{\text{grlex}} x^4yz^2$ siden graden på begge sider er 7 og $x^5yz >_{\text{lex}} x^4yz^2$. Om vi så ser på grevlex får vi også at $x^5yz >_{\text{grevlex}} x^4yz^2$, men av en annen grunn. Her er x^5yz er større fordi den minste variabelen z har lavere grad enn i x^4yz^2.

2.2 Monome sorteringsmetoder på polynomer

Dersom vi velger en monom sorteringsmetode $>$ og f er et polynom i $k[x_1, \ldots, x_n]$, så kan vi sortere monomene til f på en utvetydig måte ved hjelp av $>$. For å illustrere de ulike sorteringsmetodene nevnt over viser vi med et eksempel. Vi velger $f = 3xy^2z^2 + 2z - x^3 + 9x^2z^3 \in k[x, y, z]$ og sorterer.

i) lex $\Rightarrow f = -x^3 + 9x^2z^3 + 3xy^2z^2 + 2z$

ii) grlex $\Rightarrow f = 9x^2z^3 + 3xy^2z^2 - x^3 + 2z$

iii) grevlex $\Rightarrow f = 3xy^2z^2 + 9x^2z^3 - x^3 + 2z$

2.3 Terminologi

I denne delen av oppgaven klargjør vi notasjonen og terminologien som blir brukt videre.

Definisjon: *La $f = \sum_\alpha a_\alpha x^\alpha$ være et polynom ($\neq 0$) i $k[x_1, \ldots, x_n]$ og la $>$ være en monom sortering.*

i) *Multigraden til f er*

$$\text{multideg}(f) = \max(\alpha \in \mathbb{Z}^n_{\geq 0} : a_\alpha \neq 0)$$

(Maksimum finnes med hensyn på $>$.)

ii) *Den ledende koeffisienten til f er*

$$\text{LC}(f) = a_{\text{multideg}(f)} \in k.$$

iii) *Det ledende monomet til f er*

$$\text{LM}(f) = x^{\text{multideg}(f)}$$

(med koeffisient 1).

iv) *Det ledende leddet til f er*

$$\text{LT}(f) = \text{LC}(f) \cdot \text{LM}(f).$$

For å illustrere kan vi ta polynomet $f = 3x^2yz^3 + 5y^2 - 8x^4 - 3y^4z^3$ og la $>$ betegne lex-sortering. Vi velger $x > y > z$. Da får vi

$$\begin{aligned} \text{multideg}(f) &= (4, 0, 0), \\ \text{LC}(f) &= -8, \\ \text{LM}(f) &= x^4, \\ \text{LT}(f) &= -8x^4. \end{aligned}$$

2.4 Divisjonsalgoritme i $k[x_1, \ldots, x_n]$

I polynomringen med en variabel kjenner vi til en divisjonsalgoritme, nemlig standard polynomdivisjon. Siden vi nå befinner oss i polynomringen med flere variable, kan vi ikke bruke denne og vi må lage oss en ny divisjonsalgoritme. I tillegg vil vi gjerne dele polynomet vårt på flere polynom samtidig. I det generelle tilfellet er målet å dividere $f \in k[x_1, \ldots, x_n]$ på $f_1, \ldots, f_s \in k[x_1, \ldots, x_n]$. Det vil si at vi vil uttrykke f slik:

$$f = a_1 f_1 + \cdots + a_s f_s + r,$$

der koeffisientene $a_1, \ldots, a_s$, samt resten r, ligger i $k[x_1, \ldots, x_n]$.

Tanken bak denne algoritmen er den samme som for vanlig polynomdivisjon. Vi vil kansellere det ledende leddet til f ved å multiplisere en f_i med et passende monom og trekke produktet fra f. Dette monomet setter vi som et ledd i koeffisienten a_i.

Divisjonsalgoritmen i $k[x_1, \ldots, x_n]$. *Velg en monom sortering $>$ på $\mathbb{Z}^n_{\geq 0}$, og la $F = (f_1, \ldots, f_s)$ være et sortert s-tuppel av polynomer i $k[x_1, \ldots, x_n]$. Da kan enhver $f \in k[x_1, \ldots, x_n]$ bli skrevet som*

$$f = a_1 f_1 + \cdots + a_s f_s + r,$$

der a_i, $r \in k[x_1, \ldots, x_n]$, og hvor r enten er 0 eller en lineærkombinasjon av monomer som ikke er delelige med noen av $LT(f_1), \ldots, LT(f_s)$. Vi kaller r for resten av f dividert med F. Videre, dersom $a_i f_i \neq 0$, får vi multideg$(f) \geq$ multideg$(a_i f_i)$.

Pseudokoden for det generaliserte tilfellet ser slik ut:

input: $f_1, \ldots, f_s, f$
output: $a_1, \ldots, a_s, r$
$a_1 := 0; \ldots; a_s := 0; r := 0$
p:=f
while $p \neq 0$ **do**
  $i := 1$
  divisionoccured := false
  while ı $\leq s$ **and** divisionoccured = false **do**
    if LT(f_1) deler LT(p) **then**
      $a_i := a_i +$ LT(p)/LT(f_i)
      $p := p -$ (LT(p)/LT(f_i))f_i
      divisionoccured := true
    else
      i := i + 1
    end if
  end while
  if divisionoccured := false **then**
    $r := r +$ LT(p)
    $p := p -$ LT(p)
  end if
end while

Det må nevnes at denne algoritmen ikke gir like 'fine' resultater som vi kjenner til ved vanlig polynomdivisjon. For eksempel er ikke resten alltid unik slik den vil være i det monovariable tilfellet,

da det å endre rekkefølgen på f_i-ene påvirker resultatet i stor grad. Dersom vi for eksempel vil dele $f = x^2y + xy^2 + y^2$ på $f_1 = xy - 1$ og $f_2 = y^2 - 1$ får vi

$$x^2y + xy^2 + y^2 = (x + y) \cdot (xy - 1) + 1 \cdot (y^2 - 1) + x + y + 1$$

og resten blir $r_1 = x + y + 1$. Dersom vi bytter rekkefølge på f_1 og f_2 får vi

$$x^2y + xy^2 + y^2 = (x + 1) \cdot (y^2 - 1) + x \cdot (xy - 1) + 2x + 1.$$

Her blir resten $r_2 = 2x + 1$ og vi ser at $r_1 \neq r_2$.

En annen ulempe er at resten ikke nødvendigvis blir null selv om f er med i idealet generert av f_1 og f_2. Dersom vi lar $f_1 = xy + 1$, $f_2 = y^2 - 1 \in k[x, y]$, og deler $f = xy^2 - x$ på $F = (f_1, f_2)$, får vi

$$xy^2 - x = y \cdot (xy - 1) + 0 \cdot (y^2 - 1) + (-x - y).$$

Derimot om $F = (f_2, f_1)$ får vi

$$xy^2 - x = x \cdot (y^2 - 1) + 0 \cdot (xy + 1) + 0.$$

Vi ser av den andre utregningen at resten er 0 og $f \in \langle f_1, f_2 \rangle$, men vi ser på den første utregningen at vi kan få en rest ulik null ved divisjon på $F = (f_1, f_2)$.

Dette er problemer vi gjerne vil få bukt med og det er her Gröbner-baser kommer inn. Tanken bak er at vi vil lage en ny genererende mengde for F basert på f_1 og f_2, som gir oss de egenskapene vi vil ha.

3 Gröbner-baser

Definisjon: *Velg en monom sorteringsmetode. En endelig delmengde* $G = \{g_1, \ldots, g_t\}$ *av et ideal* I *er en Gröbner-basis dersom:*

$$\langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_t) \rangle = \langle \mathrm{LT}(I) \rangle$$

Ekvivalent, men mer uformelt, kan vi si at en mengde $\{g_1, \ldots, g_t\} \subset I$ er en Gröbner-basis til I hvis og bare hvis det ledende leddet til ethvert element i I er delelig på en eller annen $\mathrm{LT}(g_i)$.

Dette er ikke de eneste definisjonene av Gröbner-baser. Alle er ekvivalente selvsagt, men det er en annen måte som direkte tillater kalkuleringer i faktorringen R/I med de samme forutsetningene som modulær aritmetikk.

Alternativ definisjon: *En endelig delmengde* $G = (g_1, \ldots, g_s)$ *av et ideal* I *er en Gröbner-basis dersom multivariat divisjon av et vilkårlig polynom* $f \in k[x_1, \ldots, x_n]$ *med* G *gir en unik rest (uavhengig av rekkefølge på* g_i*-ene). Dersom resten blir* 0 *er* $f \in I$*.*

3.1 Egenskaper til Gröbner-baser

Nå når vi har definisjonen ser vi på noen fordeler Gröbner-baser gir oss. Gröbner-baser har flere fine egenskaper som gjør at de kan fungere som en problemløser i mange situasjoner.

Den alternative definisjonen forteller oss hvordan Gröbner-baser hjelper divisjonsalgoritmen i det multivariable tilfellet til å få en unik rest. Dermed løser det problemet om at resten til et polynom generert av idealet ikke nødvendigvis blir lik null. Legg merke til at det er kun resten som er unik, koeffisientene vi får ved divisjonsalgoritmen kan variere.

Vi ser nå på hvordan vi kan undersøke om en gitt generatormengde er en Gröbner-basis.

Definisjon: *Vi betegner resten ved deling av* f *på det sorterte* s*-tuppelet* $F = (f_1, \ldots, f_s)$*,* $\overline{f}^F$*. Dersom* F *er en Gröbner-basis, kan vi se på* F *som en mengde (uten noen spesiell sortering).*

For eksempel kan vi velge $F = (x^3y - y^2, x^2y^2 - y^2) \in k[x, y]$ og leksikografisk sortering. Da får vi $\overline{x^5y}^F = y^2$ siden divisjonsalgoritmen gir oss at $x^5y = x^2(x^3y - y^2) + 1 \cdot (x^2y^2 - y^2) + y^2$.

Det som er problemet med å se om $\{f_1, \ldots, f_s\}$ er en Gröbner-basis, er at det er mulig for polynomkombinasjoner av f_i-ene å oppstå der de ledende leddene ikke er i idealet generert av $\mathrm{LT}(f_i)$. En mulig måte dette kan skje på er at $ax^\alpha f_i - bx^\beta f_j$ kansellerer, slik at bare mindre ledd blir igjen. På den andre siden er $ax^\alpha f_i - bx^\beta f_j \in I$, så det ledende leddet er i $\langle \mathrm{LT}(I) \rangle$. For å jobbe med disse kanselleringene definerer vi følgende:

Definisjon: *La* f*,* $g \in k[x_1, \ldots, x_n]$ *være polynomer ulik null.*

(i) *Dersom* $\mathrm{multideg}(f) = \alpha$ *og* $\mathrm{multideg}(g) = \beta$*, så la* $\gamma = (\gamma_1, \ldots, \gamma_n)$*, der* $\gamma_i = \max(\alpha_i, \beta_i)$ *for hver* i*. Vi kaller* x^γ *for minste felles multiplum til* $\mathrm{LM}(f)$ *og* $\mathrm{LM}(g)$*, og skriver* $x^\gamma = \mathrm{LCM}(\mathrm{LM}(f), \mathrm{LM}(g))$*.*

(ii) *S-polynomet til* f *og* g *er*

$$S(f, g) = \frac{x^\gamma}{\mathrm{LT}(f)} \cdot f - \frac{x^\gamma}{\mathrm{LT}(g)} \cdot g$$

Vi kan se på et enkelt eksempel på utregning av et S-polynom. La $f = x^4y^3 - x^2y^3$ og $g = 4x^5y + y^3$ i $\mathbb{R}[x, y]$ med leksikografisk totalgradssortering.

$$\begin{aligned} S(f,g) &= \frac{x^5y^3}{x^4y^3} \cdot f - \frac{x^5y^3}{4x^5y} \cdot g \\ &= x \cdot f - \frac{1}{4} \cdot y^2 \cdot g \\ &= -x^3y^3 - \frac{1}{4}y^5 \end{aligned}$$

Vi konstruerer S-polynomene $S(f, g)$ for å kansellere ledende ledd. Det følgende lemmaet viser at enhver kansellering av ledende ledd i polynomer med samme multigrad, skyldes denne typen kansellering.

Lemma 3.1. *Anta at vi har $\sum_{i=1}^{s} c_i f_i$, der $c_i \in k$ og multideg$(f_i) = \delta \in \mathbb{Z}^n_{\geq 0}$ for alle i. Dersom multideg$(\sum_{i=1}^{s} c_i f_i) < \delta$ så er $\sum_{i=1}^{s} c_i f_i$ en lineærkombinasjon, med koeffisienter i k, av S-polynomet $S(f_j, f_k)$ for $1 \leq j, k \leq s$. I tillegg er multideg$(S(f_i, f_k)) < \delta$ for alle disse.*

Bevis: Se [7] □

Dersom $f_1, \ldots, f_s$ tilfredsstiller dette; får vi en ligning på formen

$$\sum_{i=1}^{s} c_i f_i = \sum_{j,k} c_{jk} S(f_j, f_k).$$

Vi ser på hvor kanselleringene skjer. I summen på venstre side så har ethvert tillegg $c_i f_i$ multigrad δ. Dette forteller oss at kanselleringen skjer etter addisjonen. På høyre side derimot, har alle tilleggene $c_{jk}S(f_j, f_k)$ multigrad mindre enn δ, så kanselleringen har allerede skjedd. Intuitivt ser vi at all kansellering skyldes S-polynomene. Ved å bruke S-polynomer og Lemma 3.1 kommer vi frem til følgende betingelse for når en generatormengde til et ideal er en Gröbner-basis:

Buchbergers kriterium. *La I være et polynomt ideal. Da er en generatormengde $G = \{g_1, \ldots, g_t\}$ for I en Gröbner-basis for I hvis og bare hvis, resten etter divisjon av $S(g_i, g_j)$ på G (vilkårlig sortert) er null.*

Bevis: $\Rightarrow$: Dersom G er en Gröbner-basis så vil vi få, siden $S(g_i, g_j) \in I$, at resten ved divisjon av G være lik null.
$\Leftarrow$(kort forklaring): La $f \in I$ være et polynom ulik null. Vi må vise at dersom alle S-polynomene har rest null modulo G, så vil $\mathrm{LT}(f) \in \langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_t) \rangle$. Gitt $f \in I = (g_1, \ldots, g_t)$ så finnes det polynomer $h_i \in k[x_1, \ldots, x_n]$ slik at

$$f = \sum_{i=1}^{t} h_i g_i.$$

Vi ser intuitivt at multideg$(f) \leq \max(\text{multideg}(h_i g_i))$. Dersom vi ikke har likhet så har det vært en form for kansellering blant de ledende leddene i uttrykket over. Lemma 3.1 medfører at vi kan skrive dette ved hjelp av S-polynomer. Da vil antagelsen vår om at S-polynomene har rest lik null medføre at vi kan erstatte S-polynomene med uttrykk med mindre kansellering. Dermed får vi et uttrykk for f med færre kanselleringer av ledende ledd. Fortsetter vi sånn vil vi tilslutt få likhet av multigrad. Da vil multideg$(f) =$ multideg$(h_i g_i)$ for en passende i og det følger at $\mathrm{LT}(g_i)$ deler $\mathrm{LT}(f)$. Dette vil da si at $\mathrm{LT}(f) \in \langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_1) \rangle$, som er det vi vil vise. □

Vi tar et raskt eksempel fra [7] på hvordan Buchbergers kriterium kan anvendes. Vi ser på idealet $I = \langle y-x^2, z-x^3\rangle$. Vi hevder at $G = \{y-x^2, z-x^3\}$ er en Gröbner-basis med leksikografisk ordning $y > z > x$. For å bevise dette ser vi på S-polynomet

$$S(y-x^2, z-x^3) = \frac{yz}{y}(y-x^2) - \frac{yz}{z}(z-x^3) = -zx^2 + yx^3$$

Bruker vi divisjonsalgoritmen finner vi så:

$$-zx^2 + yx^3 = x^3 \cdot (y-x^2) + (-x^2) \cdot (z-x^3) + 0$$

slik at $\overline{S(y-x^2, z-x^3)}^G = 0$. Buchbergers kriterium er oppfylt og dermed er G en Gröbner-basis for I. Som en kuriositet kan vi legge merke til at G ikke er en Gröbner-basis dersom sorteringen hadde vært $x > y > z$.

3.2 Buchbergers algoritme

Vi har sett på noen av Gröbner-basenes egenskaper og løst problemet om en generatormengde til et ideal faktisk er en Gröbner-basis. Det logiske fortsettelsen blir da å løse følgende problem: Gitt et ideal $I \in k[x_1, \ldots, x_n]$, finn en Gröbner-basis til I. Buchbergers kriterium leder oss på en naturlig måte i retning løsningen.

For å komme nærmere en algoritme for å finne en Gröbner-basis ser vi på et eksempel [7]. Vi ser på ringen $k[x, y]$ med leksikografisk totalgradssortering, og lar $I = \langle f_1, f_2\rangle = \langle x^3-2xy, x^2y-2y^2+x\rangle$. Her ser vi at $\{f_1, f_2\}$ ikke er en Gröbner-basis siden $\mathrm{LT}(S(f_1, f_2)) = -x^2 \notin \langle \mathrm{LT}(f_1), \mathrm{LT}(f_2)\rangle$.

En naturlig tankegang for å komme videre vil være å utvide den originale genererende mengden ved å legge til flere polynomer til I. På en måte legger vi ikke til noe nytt, det er heller et innslag av redundans, men greier vi å komme frem til en Gröbner-basis er det verdt det.

Da kommer det logiske spørsmålet om hva skal vi legge til. Svaret vårt ligger i S-polynomene. Vi vet at $S(f_1, f_2) = -x^2$ som jo er ulikt null. Derfor legger vi denne resten til vårt genererende sett, det vil si $f_3 = -x^2$. Vi setter $F = (f_1, f_2, f_3)$ og sjekker om Buchbergers kriterium er oppfylt. Vi beregner:

$$\begin{aligned}
S(f_1, f_2) &= f_3 \ \textit{og} \\
\overline{S(f_1, f_2)}^F &= 0, \\
S(f_1, f_3) &= (x^3 - 2xy) - (-x)(-x^2) = -2xy, \text{ men} \\
\overline{S(f_1, f_3)}^F &= -2xy \neq 0.
\end{aligned}$$

Siden dette ikke blir 0 er vi fremdeles ikke kommet frem til en Gröbner-basis. Vi legger inn $f_4 = -2xy$ i vårt genererende sett og prøver igjen med $F = (f_1, f_2, f_3, f_4)$.

$$\begin{aligned}
\overline{S(f_1, f_2)}^F &= \overline{S(f_1, f_3)}^F = 0, \\
S(f_1, f_4) &= y(x^3 - 2xy) - (-\frac{1}{2})x^2(-2xy) = -2xy^2 = yf_4, \ \text{ og} \\
\overline{S(f_1, f_4)}^F &= 0, \\
S(f_2, f_3) &= (x^2y - 2y^2 + x) - (-y)(-x^2) = -2y^2 + x, \ \text{ men} \\
\overline{S(f_2, f_3)}^F &= -2y^2 + x \neq 0.
\end{aligned}$$

Vi fortsetter som før og dermed blir $f_5 = -2y^2 + x$. Ved å bruke $F = \{f_1, f_2, f_3, f_4, f_5\}$ kan man kontrollere at man får:

$$\overline{S(f_i, f_j)}^F = 0 \text{ for alle } 1 \leq i \leq j \leq 5.$$

Ved Buchbergers kriterium følger det at en grlex Gröbner-basis for I er gitt ved

$$\{f_1, f_2, f_3, f_4, f_5\} = \{x^3 - 2xy, x^2y - 2y^2 + x, -x^2, -2xy, -2y^2 + x\}.$$

Eksempelet forteller oss at vi bør prøve å utvide generatormengden F til en Gröbner-basis ved å legge til rester (ulik null) $\overline{S(f_i, f_j)}^F$ til F. Dette er en naturlig konsekvens av Buchbergers kriterium og dermed har vi en algoritme for å lage Gröbner-baser.

Buchbergers algoritme: La $I = \langle f_1, \ldots, f_s \rangle \neq \{0\}$ være et polynomisk ideal. Da kan man konstruere en Gröbner-basis for I, i et endelig antall steg, ved følgende algoritme:

input: $F = (f_1, \ldots, f_s)$
output: en Gröbner-basis $G = (g_1, \ldots, g_t)$ for I, med $F \subset G$
$G := F$
repeat
  $\hat{G} := G$
  for each pair $\{p, q\}, p \neq q$ in $\hat{G}$ **do**
    $S := \overline{S(p, q)}^{\hat{G}}$
    if $S \neq 0$ **then**
      $G := G \cup \{S\}$
    end if
  end for
until $G = \hat{G}$

Grunnen til at vi kan være sikker på at algoritmen terminerer er at polynomringen $k[x_1, \ldots, x_n]$ er noethersk.

En ting vi kan bemerke er at det ikke er nødvendig å beregne $\overline{S(p, q)}^{\hat{G}}$ hver gang vi legger til et nytt element i generatormengden. Har vi først kommet frem til null, vil det forbli null.

Etter å ha jobbet litt med algoritmen ser vi at Gröbner-basisen vår som regel blir større enn nødvendig. Dette er ikke optimalt, da vi helst vil unngå redundans. For å få vekk unødvendige elementer i Gröbner-basisen bruker vi følgende lemma til å fjerne noen av generatorene.

Lemma 3.2. *La G være en Gröbner-basis til det polynome idealet I. La $p \in G$ være et polynom slik at $LT(p) \in \langle LT(G - \{p\}) \rangle$. Da er $G - \{p\}$ også en Gröbner-basis for I.*

Bevis: Vi vet at $\langle (G) \rangle = \langle (\mathrm{LT}(I) \rangle$. Dersom $\mathrm{LT}(p) \in \langle \mathrm{LT}(G - \{p\}) \rangle$, så får vi $\langle \mathrm{LT}(G - \{p\}) \rangle = \langle \mathrm{LT}(G) \rangle$. Dermed er $G - \{p\}$ ved definisjon en Gröbner-basis til I. □

Ved å tilpasse konstanter slik at alle ledende koeffisienter blir 1, og å fjerne alle p med $\mathrm{LT}(p) \in \langle (G - \{p\}) \rangle$ fra G, får vi det vi kaller en minimal Gröbner-basis.

Definisjon: *En minimal Gröbner-basis til et polynomt ideal I er en Gröbner-basis G til I slik at:*

(i) $\mathrm{LC}(p) = 1$ *for alle* $p \in G$.

(ii) *For alle* $p \in G$, $\mathrm{LT}(p) \notin \langle \mathrm{LT}(G - \{p\})\rangle$.

Vi ser på eksempelet vi gikk gjennom tidligere. Utgangspunktet var idealet $I = \langle x^3 - 2xy, x^2y - 2y^2 + x\rangle$ noe som medførte 5 genererende elementer; $f_1 = x^3 - 2xy$, $f_2 = x^2y - 2y^2 + x$, $f_3 = -x^2$, $f_4 = -2xy$ og $f_5 = -2y^2 + x$. Siden enkelte av de ledende koeffisientene er ulik 1 begynner vi med å multiplisere generatorene med passende konstanter slik at de får koeffisient 1. I praksis vil det si at vi ser på de ledende monomene til generatorelementene. Vi ser at $\mathrm{LT}(f_1) = x^3 = x \cdot \mathrm{LT}(f_3)$. Dermed kan vi bruke Lemma 3.2 og fjerne f_1 fra generatormengden vår. På samme måte kan vi fjerne f_2 ved hjelp av f_4. Da er det ikke lenger noen eliminasjonsmuligheter og vi sitter igjen med en minimal Gröbner-basis $\tilde{f}_3 = x^2$, $\tilde{f}_4 = xy$ og $\tilde{f}_5 = y^2 - \frac{1}{2}x$.

Dessverre er ikke denne Gröbner-basisen unik. Et gitt ideal kan ha flere ulike minimale Gröbner-baser. I eksempelet over er det lett å sjekke at $\tilde{f}_3 = x^2 + axy$, $\tilde{f}_4 = xy$ og $\tilde{f}_5 = y^2 - \frac{1}{2}x$ også vil være en minimal Gröbner-basis for en vilkårlig $a \in k$. Så hvilken av disse er best? Vi definerer redusert Gröbner-basis.

Definisjon: *En redusert Gröbner-basis til en polynomt ideal* I *er en Gröbner-basis* G *til* I *slik at:*

(i) $\mathrm{LC}(p) = 1$ *for alle* $p \in G$.

(ii) *For alle* $p \in G$ *vil ingen monomer i* p *ligge i* $\langle \mathrm{LT}(G - \{p\})\rangle$.

I eksempelet over ser vi at den eneste reduserte Gröbner-basisen vil ha $a = 0$. Generelt sett har reduserte Gröbner-baser følgende fine egenskap:

Proposisjon 3.3. *La* $I \neq \{0\}$ *være et polynomisk ideal. Da har, for en gitt monom sortering,* I *en unik, redusert Gröbner-basis.*

Bevis: Se [7] □

En konsekvens av unikheten er at vi kan lage en algoritme som undersøker om to ulike polynommengder $\{f_1, \ldots, f_s\}$ og $\{g_1, \ldots, g_t\}$ genererer det samme idealet. Vi velger en monom sortering og beregner en redusert Gröbner-basis for mengdene. Idealene vil være like hvis og bare hvis de reduserte Gröbner-basene er like.

3.2.1 Buchbergers algoritme og radredusering

Avslutningsvis i Buchbergers algoritme-delen skal vi se nærmere på en fascinerende sammenheng mellom denne algoritmen og vanlig gaussisk eliminasjon av lineære ligninger. Det viser seg at vi kan se på den radreduserende algoritmen et spesialtilfelle av Buchbergers algoritme. For å få frem enkeltheten bruker vi også her et eksempel fra [7]:

$$\begin{aligned} 3x - 6y - 2z &= 0, \\ 2x - 4y + 4w &= 0, \\ x - 2y - z - w &= 0. \end{aligned}$$

Vi bruker radoperasjoner på koeffisientmatrisen, setter opp på echelonform og får matrisen:

$$\begin{pmatrix} 1 & -2 & -1 & -1 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Vi finner den reduserte echelonformen:

$$\begin{pmatrix} 1 & -2 & 0 & 2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

For å oversette disse beregningene til algebra, lar vi I være idealet

$$I = \langle 3x - 6y - 2z, 2x - 4y + 4w, x - 2y - z - w \rangle \subset k[x, y, z, w]$$

som tilsvarer det originale ligningssystemet. Vi bruker lex-ordning med $x > y > z > w$. Dersom vi undersøker dette idealet finner vi at de lineære ligningene bestemt av den øverste echelonmatrisen gir oss en minimal Gröbner-basis

$$I = \langle x - 2y - z - w, z + 3w \rangle,$$

og videre finner vi at den reduserte echelonmatrisen gir oss den reduserte Gröbner-basisen

$$I = \langle x - 2y + 2w, z + 3w \rangle.$$

Vi vet fra lineær algebra at enhver matrise kan reduseres til redusert echelonform på en unik måte. Dette kan vi sammenligne med unikheten til reduserte Gröbner-baser.

3.3 Bruk av Gröbner-baser

3.3.1 Løsning av ligningssystem

Kanskje det mest aktuelle bruken av Gröbner-baser for dette prosjektet ligger i hvordan vi kan bruke Gröbner-baser til å løse multivariable ligningssystemer. Vi ser på ligningene:

$$\begin{aligned} x^2 + y^2 + z^2 &= 1, \\ x^2 + y^2 &= y, \\ x &= z, \end{aligned}$$

i $\mathbb{C}^3$. Disse ligningene gir oss idealet $I = \langle x^2 + y^2 + z^2 - 1, x^2 + z^2 - y, x - z \rangle \subset \mathbb{C}[x, y, z]$. Vi vil finne $V(I)$, det vil si alle punkter (x, y, z) som oppfyller ligningssettet. Teoretisk sett kan vi beregne $V(I)$ ut fra en vilkårlig generatormengde til I, men det er ikke alltid så lett i praksis. Hva skjer så om vi benytter en Gröbner-basis i utregningen?

Vi beregner en Gröbner-basis med lex-ordning. Denne blir her $g_1 = x - z$, $g_2 = -y + 2z^2$ og $g_3 = z^4 + \frac{1}{2}z^2 - \frac{1}{4}$. Ser vi nærmere på disse polynomene ser vi at g_3 bare er avhengig av z. Dette medfører at

$$z = \pm\frac{1}{2}\sqrt{\pm\sqrt{5} - 1},$$

som gir oss 4 mulige verdier for z. Setter vi disse verdiene inn i $g_1 = 0$ og $g_2 = 0$ finner vi de unike verdiene for x og y. Dermed ser vi at det finnes fire løsninger, to reelle og to komplekse. Dette er alle mulige løsninger av de opprinnelige ligningene.

Hvis vi prøver den samme fremgangsmåten på andre eksempler skjer akkurat det samme. Vi får ligninger der variablene blir eliminert og systemet blir enkelt å løse. I tillegg ser vi at det er de samme variable som blir eliminert først hver gang. I eksempelet over hadde vi leksikografisk ordning og vi ser at den siste ligningen bare var avhengig av den minste variabelen. Dette viser seg å stemme universelt.

3.3.2 Idealmedlemskapsproblemet

Et annet problem vi gjerne vil ha svar på er om et gitt polynom f er med i idealet vårt I. For å besvare dette spørsmålet kombinerer vi Gröbner-basene med divisjonsalgoritmen, og får det vi kaller idealmedlemskapsalgoritmen. Dette medfører at for et gitt ideal, $I = \langle f_1, \ldots, f_s \rangle$, kan vi finne ut om f ligger i I på følgende måte: Først beregner vi en Gröbner-basis $G = \{g_1, \ldots, g_t\}$ til I ved Buchbergers algoritme. Dermed vet vi at

$$f \in I \iff \overline{(f)}^G = 0.$$

For å klargjøre dette ser vi nærmere på et eksempel. Vi lar $I = \langle f_1, f_2 \rangle = \langle xz - y^2, x^3 - z^2 \rangle \in \mathbb{C}[x, y, z]$ og velger leksikografisk totalgradssortering. La $f = -4x^2y^2z^2 + y^6 + 3z^5$. Er da f i I?

Vi ser at den genererende mengden her ikke er en Gröbner-basis siden $\mathrm{LT}(I)$ inneholder polynomer som $(\mathrm{LT}(S(f_1, f_2)) = \mathrm{LT}(-x^2y^2 + z^3) = -x^2y^2$ som ikke er i idealet $\langle xz, x^3 \rangle$. Vi bruker Buchbergers algoritme til å finne en Gröbner-basis og reduserer denne. Dermed ender vi opp med den reduserte Gröbner-basisen

$$G = \{f_1, f_2, f_3, f_4, f_5\} = \{xz - y^2, x^3 - z^2, x^2y^2 - z^3, xy^4 - z^4, y^6 - z^5\}.$$

Nå kan vi teste polynomer om de er med i I. Vi sjekker $f = -4x^2y^2z^2 + y^6 + 3z^5$ som ble nevnt lenger oppe. Vi deler på G og får

$$f = (-4xy^2z - 4y^2) \cdot f_1 + (-3) \cdot f_4 + 0$$

Vi ser at resten er null og dette impliserer $f \in I$.

Det er forøvrig ikke alltid disse beregningene er nødvendig. Av og til kan vi se utfallet direkte. Dersom vi for eksempel vil sjekke $f = xy - yz + 2z^2 + x$ kan vi se direkte av elementene i G at $f \notin G$. Dette fordi $\mathrm{LT}(f) = xy$ åpenbart ikke er i idealet $\langle \mathrm{LT}(G) \rangle = \langle xz, x^3, x^2y^2, xy^4, y^6 \rangle$. Dermed er $\overline{f}^G \notin 0$ og vi får $f \notin I$.

Dette viser oss hvordan idealets egenskaper gjenspeiles i elementene til Gröbner-basisen.

4 Problemene med kryptosystem basert på Gröbner-baser

Egenskapene til Gröbner-baser får oss til å lure på om ikke disse kan brukes i krypteringsøyemed. Siden det tar *lang* tid å finne en Gröbner-basis, burde det kunne være mulig å lage et offentlig nøkkel-kryptosystem ut fra dette. Det viser seg imidlertid at det ikke er så lett som først antatt. I 1994 ble artikkelen *Why you cannot even hope to use Gröbner bases in Public Key Cryptography: an open letter to a scientist who failed and challenge to those who have not yet failed* av B. Barkee et al. [1] utgitt. I artikkelen setter forfatterne fingeren på en del av problemene med disse kryptosystemene.

Vi ser på et ideal $I \subset k[x_1, \ldots, x_n]$ og en sorteringsmetode på semigruppen T, bestående av de moniske monomene i $k[x_1, \ldots, x_n]$. Sorteringen tillater oss å skrive ethvert polynom $f \in k[x_1, \ldots, x_n]$ som en unik, sortert lineærkombinasjon av elementene i T. Videre gir ordningen oss muligheten til å knytte idealet I til semigruppeidealet $T(I) := \{T(f) \in T : f \in I \backslash \{0\}\} \subset T$ samt det komplementære ordensidealet $O(I) := T \backslash T(I)$. Da ser vi følgende:

(i) $k[x_1, \ldots, x_n] = I \bigoplus \mathrm{Span}_k(O(I))$.

(ii) Det finnes et k-vektorromisomorfi mellom $k[x_1, \ldots, x_n]/I$ og $\mathrm{Span}_k(O(I))$.

(iii) For hver $f \in k[x_1 \ldots, x_n]$ finnes det en unik $g := \mathrm{Can}(f, I) \in \mathrm{Span}_k(O(I))$ slik at $f - g \in I$.

(iv) $\mathrm{Can}(f, I) = \mathrm{Can}(g, I) \Longleftrightarrow f - g \in I$.

(v) $\mathrm{Can}(f, I) = 0 \Longleftrightarrow f \in I$.

Her påpeker forfatterne et viktig poeng som ofte oversees i presentasjoner av Gröbner-baser: Den kanoniske formen til et element defineres bare ved hjelp av leddene i idealet og sorteringen. Konseptuelt trenger vi ikke Gröbner-baser for å definere kanoniske former. Videre ser vi på hvordan Gröbner-baser kan hjelpe i denne situasjonen.

Dersom en Gröbner-basis G til I er kjent og vi har $f \in k[x_1, \ldots, x_n]$, så kan den kanoniske formen $\mathrm{Can}(f, I)$ beregnes ved følgende algoritme (Buchbergers redusering):

$Red(f; G)$
$h := 0$
while $f \neq 0$ **do**
  if $T(f) \in T(G)$ **then**
    velg $g \in G$ slik at $T(f) = tT(g)$
    $f := f - \mathrm{LC}(f)\mathrm{LC}(g)^{-1}tg$
  else
    $h := h + \mathrm{LC}(f)T(f),\ f = f - \mathrm{LC}(f)T(f)$‘
  end if
end while
$\mathrm{Can}(f, I) := h$

Denne algoritmen er analog til gaussisk reduksjon over vektorrom og kjøretiden blir derfor kvadratisk med tanke på størrelsen til input-polynomet f. Dersom sorteringen tar hensyn til graden til f, det vil si $\deg(t_1) < \deg(t_2) \Longrightarrow t_1 < t_2$, vil kjøretiden bli $O(d^n)$, der $d = \deg(f)$. Dette kan sammenlignes med Buchbergers algoritme som finner en Gröbner-basis til et ideal I, gitt en basis $\{f_1, \ldots, f_t\}$. Her er worst case-kompleksiteten $d^{2^{O(n)}}$, der d er den høyeste graden av f_i-ene.

Selv uten kompleksitetsresultatene, så er den vanlige antagelsen om at Gröbner-baser er vanskelige å beregne basert på en misforståelse. Det stemmer at det finnes idealer med Gröbner-baser som

har elementer hvis grad er dobbelt eksponentiell i graden til input-basisen, men slike eksempler må spesialvelges og er overhodet ikke tilfeldige. Dette strider mot et viktig krypteringsprinsipp; vi *må* ha tilfeldighet i krypteringen vår.

I tillegg blir det følgende bevist i en av artiklene som delvis finner kompleksiteten til Buchbergers algoritme[16]:

Teorem 4.1. De fleste *ideal generert av s polynomer i n variable av grad bundet av d er slik at deres Gröbner-baser har grad bundet av* $(n+1)d-n$.

Med *de fleste* menes det at koeffisientene er tilfeldig valgt og at resultatet holder, unntatt for en mengde med mål null i rommet som parametriserer koeffisientene.

En annen vesentlig misforståelse er å sammenblande idealmedlemskapsproblemet med problemet å finne en Gröbner-basis. En løsning på det ene problemet, løser det andre problemet, men det kan finnes enklere måter å løse det første problemet på. Det er denne enklere løsningen som kan brukes til å knekke et Gröbner-kryptosystem.

Et enkelt kryptosystem basert på Gröbner-baser

Det følgende er den enkleste måten å se for seg et kryptosystem basert på Gröbner-baser. Denne ideen, og varianter av den, har dukket opp mange steder, men har aldri blitt gjennomført etter at forfatterne forklarte angrepet som vil bli forklart senere. [1]

Alice offentliggjør en mengde ledd $L \subset O(I)$ (enten hele $O(I)$ eller, for ekstra sikkerhet, en undermengde av den) og en gruppe polynomer $\{g_1 \ldots, g_l\} \in I$ av lav grad, som er en basis for enten I eller et ideal ekte inneholdt i I. Når Bob vil sende en melding til Alice, krypterer han den som en lineærkombinasjon $M = \sum_{t_i \in L} c_i t_i$. Polynomet tilfredsstiller dermed $M = \mathrm{Can}(M, I)$. For å kryptere meldingen produserer så Bob tilfeldige polynomer $p_1, \ldots, p_l$ og sender polynomet $C := M + \sum_{i=1}^{l} p_i g_1$.

Siden $C - M \in I$, så vil $\mathrm{Can}(C, I) = \mathrm{Can}(M, I) = M$, så Alice kan bruke sin hemmelige Gröbner-basis til å beregne $M = \mathrm{Can}(C, I)$. Det angriperen Eve vet er da L, g_i-ene og C, men hun vet også at M, riktignok ukjent, er den kanoniske formen til C. Før vi ser nærmere på angrepet antar vi at Alice offentliggjør det følgende:

(i) l polynomer av grad høyest d, i n variable: $g_1, \ldots, g_l$.

(ii) en mengde monomer $L = \{m_1, \ldots, m_s\} \subset O(I)$ av grad høyst d.

For å sende en melding velger Bob hver p_i til å være et polynom av grad r. Da er C et polynom av grad $R \leq D := d + r$. Om vi så lar τ representere antallet ledd av grad høyst D, så vil kompleksiteten til Bobs kryptering og Alices dekryptering være henholdsvis $O(\tau)$ og $O(\tau^2)$.

Verdien τ er sannsynligvis en offentlig parameter, men Eve vil ikke hindres nevneverdig i angrepet sitt om denne verdien hadde vært Bobs hemmelige valg. Dette skyldes at det er få kanselleringer i summen $\sum_{i=1}^{l} p_i g_1$ og Eve kan få et godt estimat av τ.

Eve har nemlig en annen og sterkere fordel som skyldes unikheten til den kanoniske formen. Eve trenger ikke finne de samme p_i-ene som Bob brukte, ethvert valg av q_i-er slik at $C = M + \sum_{i=1}^{l} q_i g_i$ fungerer like bra for henne. Spesifikt kan Eve søke etter minimalgradsrepresentasjonen $C = M + \sum_{i=1}^{l} q_i g_i$. Setter vi den høyeste graden blant $q_i g_i$ lik D' får vi $R \leq D' \leq D = d + r$. På grunn av det tilfeldige elementet forventer vi riktignok at $R = D' = D$.

For å knekke dette systemet kan Eve bruke følgende resultat om Gröbner-baser [8]:

Teorem 4.2. *La $I = (g_1, \ldots, g_l)$ og la h være slik at $deg(h) \leq D, h - Can(h, I) = \sum_{i=1}^{l} p_i g_i$ med $deg(p_i g_i) \leq D$. La G være outputen fra Buchbergers algoritme modifisert slik at hver beregning med polynomer av grad høyere enn D ikke utføres. Da kan $Can(h, I)$ beregnes ved Buchbergers redusering av h via G.*

Som en konsekvens av dette kan Eve beregne en Gröbner-basis til I som utsetter alle beregninger med polynomer av grad høyere enn $R = \deg(C)$, og dermed få en mengde G. Videre beregner hun $h := Red(C, G)$. Dersom $h \in \mathrm{Span}_k(L) \subset O(I)$, så er $h = \mathrm{Can}(C, I) = M$, siden $C - h \in I$ og den kanoniske formen er unik.

Ellers vet Eve at gjetningen $R = \deg(C)$ for D' var for lav. Hun fortsetter så med å prøve $D' = R + 1$. Hun trenger selvsagt ikke å starte på nytt i Gröbner-basisutregningen, men bare å utføre de beregningene av grad $R + 1$ som ble utsatt fra den første gjennomkjøringen, sammen med eventuelle nye beregninger av grad lavere enn $R + 1$. Etter den andre gjennomkjøringen vil hun i tillegg ikke redusere C, men h.

Vi ser at algoritmen over gir Eve kunnskap om hvilken del av Gröbner-basisen som er nødvendig for å dekryptere enhver mulig melding. I artikkelen viser forfatterne at kompleksiteten er $O(\tau^4)$ [1].

På grunn av dette vil utregningene til Eve være nøyaktig de samme som om hun hadde kjent til verdien av D' og kjørte algoritmen én gang, uten beregninger av grad høyere enn D'.

Et annet angrep som også påpekes bruker bare god gammeldags lineær algebra, og det med kubisk kompleksitet. Det viktigste med dette angrepet viser åpenbart at knytningen mellom Gröbner-baser og idealmedlemskapsproblemet er mye svakere enn først antatt.

I dette tilfellet estimerer Eve D' noenlunde og velger seg verdien R. Hun løser så ligningen $M + \sum_{i=1}^{l} q_i g_i = C$, der q_i-ene er polynomer av grad $R - \deg(g_i)$ med ukjente koeffisienter. Dette er et lineært ligningssystem der de ukjente er koeffisientene til M og q_i-ene og hvis ligninger bare er koeffisienter til hvert ledd i polynomet $M + \sum_{i=1}^{l} q_i g_i - C = 0$. Løser man så dette får vi to muligheter:

(i) Det finnes løsninger, selv om det finnes mer enn én løsning så vil koeffisientene til M være bestemt og unike og Eve kan lese meldingen.

(ii) Det finnes ingen løsninger, estimatet av D' er feil. Eve prøver igjen med en høyere verdi. Siden den første undermengden av ligninger er den samme som før, trenger hun ikke gjøre alt arbeidet på nytt. Derfor, på samme måte som angrepet over, øker ikke dette kompleksiteten som er $O(\tau^3)$.

Selv om vi har sett hvordan Eve kan dekryptere en enkel beskjed, så ser vi at matriseinversjon lar henne knekke systemet på kubisk tid, så å unngå Gröbner-baser gir ingen ulemper.

Videre kommer vi til utfordringen til forfatterne. Begge angrepene over er korrekte og effektive for det vanlige Gröbner-kryptosystemet beskrevet over. De underliggende ideene er at vi bare trenger en delvis Gröbner-basis for å løse et bundet idealmedlemskapsproblem og at vi faktisk til og med kan unngå Gröbner-baser ved å ty til lineær algebra.

Den høye kompleksiteten til Gröbner-baser er faktisk strengt knyttet til eksistensen av polynomer i et ideal hvis minste gradrepresentasjon i en gitt basis er dobbeleksponentiell i graden til basiselementene. Siden slike polynomer ikke kan bli brukt som krypterte meldinger vil et kryptosystem som bruker kompleksiteten til Gröbner-baser på et idealmedlemskapsproblem nødvendigvis feile.

Utfordringen er: kan noen finne et kryptosystem som overkommer disse vanskelighetene?

Så langt har svaret vist seg å være nei. Foreløpig er det ingen kryptosystemer vi er klar over som har overvunnet disse problemene etter nøye testing. Riktignok er det noen som ikke har blitt testet i stor nok grad enda som unngår eldre angrep og som kan virke lovende. Anbefalt lesning her er for eksempel [5] som foreslår to nye kryptosystemer som blant annet baserer seg på latticer.

5 Unbalanced Oil and Vinegar

5.1 Multivariat kryptografi

Selv om det ikke er så lett å lage kryptosystemer basert på Gröbner-baser betyr det ikke at det er umulig å lage kryptosystemer ved å ta den multivariable polynomringen til hjelp. Det generelle navnet for asymmetrisk kryptografi med utgangspunkt i multivariate polynomer over endelig kropper er multivariat kryptografi. I enkelte tilfeller er polynomene definert over både en grunnkropp og en kroppsutvidelse. Dersom graden til polynomene aldri overstiger to snakker vi om multivariable kvadratiske systemer. Det å løse multivariate ligningssystem er bevist NP-komplett [12]. Per dags dato har vi ingen sikre krypteringsskjemaer av denne typen. Mange forsøk er gjort og flere ting er foreslått, men foreløpig er vi av sikkerhetsgrunner begrenset til signaturskjemaer.

Det hele begynte på slutten av 1980-tallet da T.Matsumoto og H. Imai presenterte sin variant på Eurocrypt-konferansen i 1988 [20]. Dette la grunnlaget for videre arbeid og spesielt Jacques Patarin var ivrig på området. Han utviklet ulike kryptosystem og hans 'Hidden Field Equations' ble presentert i 1996 [22]. Han stoppet ikke der og fortsatte utviklingen av andre system. I 1997 la han frem signaturskjemaet 'Balanced Oil and Vinegar' [23]. Det viste seg at denne ikke var helt trygg mot alle former for angrep, så Patarin jobbet videre sammen med Aviad Kipnis og Louis Goubin med å forbedre dette skjemaet til å stå imot angrepene som var avgjørende mot 'Balanced Oil and Vinegar'. I 1999 presenterte de 'Unbalanced Oil and Vinegar' [18]. Forskjellene mellom disse to variantene ligger i forholdet mellom de såkalte oil- og vinegar-variablene.

5.2 Generelt om signaturskjema

Meldinger som blir sendt gjennom en åpen linje kan lett bli forfalsket. Da er det viktig at man har en måte å verifisere at det er riktig avsender som står bak meldingen og det er her signaturskjemaene kommer inn i bildet. Et signaturskjema er en form for asymmetrisk kryptografi som gir mottaker en grunn til å tro at meldingen kommer fra riktig avsender. På mange måter er disse skjemaene det samme som en håndskrevet signatur, bare at en digital signatur gjerne er vanskeligere å forfalske. Slike digitale signaturer kan være hva som helst som kan bli representert med en bit-streng. Av eksempler kan vi nevne elektronisk post, kontrakter eller beskjeder sendt via en annen kryptografisk protokoll.

Definisjon: *Et signaturskjema består typisk av tre algoritmer:*

(i) En nøkkelgenererende algoritme som velger en privat nøkkel tilfeldig fra en mengde mulige private nøkler. Denne algoritmen gir oss den private nøkkelen og den tilhørende offentlige nøkkelen.

(ii) En signeringsalgoritme som, gitt en beskjed og en privat nøkkel, gir oss en signatur.

(iii) En signaturverifikasjonsalgoritme som gitt en beskjed, offentlig nøkkel og signatur, enten aksepterer eller forkaster.

Det er hovedsaklig to viktige egenskaper for at et skjema skal kunne brukes. For det første må en signatur generert fra en spesifikk melding og en spesifikk privat nøkkel kunne bli verifisert basert på nevnte melding og den korresponderende offentlige nøkkelen. For det andre må det være ugjennomførbart for en inntrenger å regne seg frem til en gyldig signatur uten å kjenne den private nøkkelen.

5.3 Motivasjon

Signaturskjemaer har med tiden blitt en viktig del av teknologien for å holde internett og andre IT-infrastrukturer sikre. Digitale signaturer brukes i det meste av identifiserings- og autentiseringsprotokoller for eksempel om du laster ned software. Derfor er sikkerheten til signaturskjemaene ytterst viktig for å opprettholde IT-sikkerheten.

Et mulig problem med dette oppstod i 1994. Peter Shor viste at kvantedatamaskiner kan knekke alle signaturskjemaene som blir brukt i praksis den dag i dag [24]. Det vanligste signaturskjemaet for eksempel, RSA, baseres på at det å primtallsfaktorisere produktet av to store primtall tar veldig lang tid, men Shors algoritme viser at dette er mulig å få til på polynomisk tid på en kvantedatamaskin. I 2001 ble algoritmen demonstrert av Issac Chuang og andre folk på IBM, som faktoriserte 15 til $3 \cdot 5$ ved implementering på en kvantedatamaskin med 7 såkalte qubits [25]. Noen fysikere spår at man i løpet av de neste 20 årene vil ha laget store nok kvantedatamaskiner til å knekke alle signaturskjemaer og kryptosystemer i bruk i dag, men ingenting har skjedd på denne fronten siden 2001 så de lærde strides.

Spørsmålet blir da hva skal vi ta i bruk hvis noen lager en vellykket kvantedatamaskin? Finnes det signaturskjemaer som er kvantesikre? Vet vi nok om deres generelle sikkerhet og effektivitet?

Per dags dato er vi langt unna å kunne erstatte eksisterende signaturskjema med nye som er sikre mot kvantedatamaskinangrep. Vi må utvikle og forske på nye metoder for å holde sikkerheten inntakt. Vi må identifisere problemer som fremdeles er vanskelige på en kvantedatamaskin og bruke disse i kryptosystemene og signaturskjemaene våre.

Denne oppgaven ser nærmere på et signaturskjema som forhåpentligvis vil være kvanteresistent. Foreløpig kjenner vi ikke til noen kvantealgoritme som vil korte ned kjøretiden på problemet som brukes her.

5.4 Unbalanced Oil and Vinegar

La $K = F_q$ være en endelig kropp (for eksempel $K = F_2$). La n og v være to tall. Meldingen som skal signeres representeres av et element i K^n, og vi kaller denne for $y = (y_1, \ldots, y_n)$. Typisk er $q^n = 2^{128}$. Signaturen er representert som et element i K^{n+v} som vi kaller $x = (x_1, \ldots, x_{n+v})$.

5.4.1 Hemmelig nøkkel

Den hemmelige nøkkelen består av to deler:

I) En bijektiv og affin funksjon $s : K^{n+v} \to K^{n+v}$. Med affin mener vi at enhver komponent i outputen kan skrives som et polynom av grad én i $n+v$ ukjente, med koeffisienter i K.

II) En mengde (S) med n ligninger på følgende form:

$$(S) \qquad \forall i,\ 1 \leq i \leq n,\ y_i = \sum \gamma_{ijk} a_j \hat{a}_k + \sum \lambda_{ijk} \hat{a}_j \hat{a}_k + \sum \xi_{ij} a_j + \sum \hat{\xi}_{ij} \hat{a}_j + \delta_i$$

Koeffisientene $\gamma_{ijk}, \lambda_{ijk}, \xi_{ij}, \hat{\xi}_{ij}$ og δ_i er de hemmelige koeffisientene til disse n ligningene. Verdiene $a_1, \ldots, a_n$ (oil-ukjente) og $\hat{a}_1, \ldots, \hat{a}_v$ (vinegar-ukjente) ligger i K. Legg merke til at S ikke inneholder noen ledd med $a_i a_j$.

5.4.2 Offentlig nøkkel

La A være elementet i K^{n+v} definert ved $A = (a_1, \ldots, a_n, \hat{a}_1, \ldots, \hat{a}_v)$. Da vil A bli transformert til $x = s^{-1}(A)$, der s er den hemmelige, bijektive og affine funksjonen fra K^{n+v} til K^{n+v}. Hver verdi

$y_i, 1 \leq i \leq n$, kan skrives som et polynom P_i av total grad to i de x_j ukjente, $1 \leq j \leq n+v$. Vi skriver (P) for mengden av de følgende n ligningene:

$$(P) \qquad \forall i,\ 1 \leq i \leq n,\ y_i = P_i(x_1, \ldots, x_{n+v})$$

Disse n kvadratiske ligningene er den offentlige nøkkelen.

5.4.3 Beregning av signatur (med hemmelig nøkkel)

Beregning av signaturen x til meldingen y gjøres på følgende måte:

Først finner vi n ukjente $a_1, \ldots, a_n$ i K og v ukjente $\hat{a}_1, \ldots, \hat{a}_v$ i K slik at alle n ligningene (S) er tilfredsstilt. Dette kan gjøres ved at vi velger tilfeldig v vinegar-ukjente $\hat{a}_i, \ldots, \hat{a}_v$ før vi beregner de n oil-ukjente $a_i, \ldots, a_n$ basert på (S) ved gaussisk reduksjon. Siden vi ikke har noen $a_i a_j$-ledd er (S)-ligningene lineære med hensyn på a_i-ene når $\hat{a}_i$-ene er fiksert. Etter at vi har gjort dette beregner vi $x = s^{-1}(A)$ der $A = \{a_1, \ldots, a_n, \hat{a}_1, \ldots, \hat{a}_v\}$. Da er x signaturen til y.

Merk at man ikke alltid vil lykkes med å lage et tilfredsstillende ligningssett etter tilfeldig valg av vinegar-ukjente. Dersom dette ikke går velger man bare på nytt. Selv etter få forsøk er sannsynligheten høy for at det skal gå minst en gang. (Dersom vi befinner oss i F_2 er sannsynligheten cirka 30% for suksess på første forsøk [18].)

5.4.4 Offentlig verifisering av en signatur

En signatur x til y godkjennes hvis og bare hvis alle polynomene i (P) er tilfredsstilt. Som en følge av dette er det ikke nødvendig med en hemmelig måte å sjekke om signaturen er riktig, og dette er derfor et asymmetrisk signaturskjema.

Navnet 'Oil and Vinegar' skyldes det at de oil-ukjente a_i og de vinegar-ukjente $\hat{a}_i$ ikke kombineres i alle muligheter i ligningene (S) siden det ikke er noen $a_i a_j$-produkter. Derimot om vi ser på (P) så skjules denne egenskapen av den affine transformasjonen s og det er nettopp her sikkerheten til UOV ligger.

5.5 Eksempel på UOV

For å illustrere tankegangen bak dette signaturskjemaet ser vi på et raskt eksempel. For at det ikke skal bli altfor uoversiktlig velger vi å jobbe over kroppen med to elementer og setter $n = 3$ og $v = 4$. Så velger vi ligningene våre i (S). For å få det på en mer oversiktlig form velger vi ikke altfor mange koeffisienter ulik null.

$$(S) \qquad \begin{aligned} y_1 &= a_1\hat{a}_2 + a_3\hat{a}_4 + \hat{a}_1\hat{a}_3 + \hat{a}_2\hat{a}_4 + a_2 + \hat{a}_1 \\ y_2 &= a_2\hat{a}_1 + a_3\hat{a}_3 + \hat{a}_2\hat{a}_4 + a_2 + \hat{a}_3 + 1 \\ y_3 &= a_1\hat{a}_1 + a_2\hat{a}_4 + \hat{a}_1\hat{a}_2 + a_1 + a_2 + \hat{a}_4 + 1. \end{aligned}$$

Vi definerer den affine transformasjonen $s : K^7 \mapsto K^7$ ved $(c_1, c_2, c_3, c_4, c_5, c_6, c_7) \mapsto (c_7, c_1, c_2, c_3, c_4, c_5, c_6)$. Så velger vi tilfeldige verdier (i $\mathbb{Z}_2$) for $\hat{a}_i$-ene våre for eksempel $\hat{a}_1 = 1, \hat{a}_2 = 1, \hat{a}_3 = 1$ og $\hat{a}_4 = 0$. Da ender vi opp med følgende ligninger:

$$\begin{aligned} y_1 &= a_1 + 0 + 1 + 0 + a_2 + 1 \\ y_2 &= a_2 + a_3 + 0 + a_2 + 1 + 1 \\ y_3 &= a_1 + 0 + 1 + a_1 + a_2 + 0 + 1. \end{aligned}$$

Som vi ser får vi et ganske enkelt ligningssystem og a_i-ene bestemmes av meldingen y som skal signeres.

La nå $y = (1, 1, 0)$. Setter vi dette inn får vi $a_1 = 1, a_2 = 0$ og $a_3 = 1$. Dermed blir vår $A = (1, 0, 1, 1, 1, 1, 0)$. For å få signaturen vår tar vi $s^{-1}(A)$. Siden vi benytter s^{-1} vil dette tilsi et venstreskift. Dermed får vi signaturen $x = (0, 1, 1, 1, 1, 0, 1)$.

Videre må vi få laget den offentlige nøkkelen. Det at vi har en signatur er tross alt irrelevant om det ikke er en måte å sjekke denne. For å lage ligningsmengden (P) tar vi utgangspunkt i den hemmelige polynommengden (S) og transformasjonen s. Vi skriver opp (P) først da forklaringen etterpå blir enklere å forstå.

$$(P) \quad \begin{aligned} P_1 &= x_2x_6 + x_4x_1 + x_5x_7 + x_6x_1 + x_3 + x_5 \\ P_2 &= x_3x_5 + x_4x_7 + x_6x_1 + x_3 + x_7 + 1 \\ P_3 &= x_2x_5 + x_3x_1 + x_5x_6 + x_2 + x_3 + x_1 + 1. \end{aligned}$$

Måten vi kom frem til disse ligningene på var som følger. Vi skriver om $A = (a_1, a_2, a_3, \hat{a}_1, \hat{a}_2, \hat{a}_3, \hat{a}_4)$ til $A_x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$. Så tar vi $s^{-1}(A_x) = (x_2, x_3, x_4, x_5, x_6, x_7, x_1) = X$. Så skriver vi om ligningene i (S) ved å bytte ut elementene i A med det korresponderende elementet i X. Det vil si at vi bytter a_1 med x_2, a_2 med x_3 og så videre. Følger man dette slavisk får man det offentlige ligningssystemet over.

Vi antar at vi mottok meldingen $y = (1, 1, 0)$ med signaturen $x = (0, 1, 1, 1, 1, 0, 1)$. Vi setter inn verdiene fra signaturen i de offentlige ligningene. (Første verdi i signatur settes inn for x_1 og så videre.) Vi får $P_1 = 1, P_2 = 1$ og $P_3 = 0$. Dette stemmer overens med meldingen vi mottok og vi aksepterer derfor at det er riktig avsender.

5.6 Angrep på UOV

Vi skal nå se nærmere på noen angrep på dette signaturskjemaet. Utgangspunktet vårt her er arbeidene til Braeken, Wolf og Preneel [3], samt Kipnis og Shamir [17] . De ser nærmere på tre ulike angrep og det første var angrepet som knakk den balanserte versjonen av Oil and Vinegar.

5.6.1 Kipnis- og Shamir-angrep

Tanken bak dette angrepet er som følger: For å skille oil-variablene fra vinegar-variablene ser vi kun på de kvadratiske leddene i de n offentlige ligningene i (P). La G_i , $1 \leq i \leq n$, være den respektive matrisen til kvadratuttrykket til P_i fra de offentlige ligningene (P). Den kvadratiske delen av ligningene i mengden (S) representeres på en kvadratisk form med en korresponderende $2n \times 2n$-matrise på formen: $\begin{pmatrix} 0 & A \\ B & C \end{pmatrix}$. Det at vi får 0 i $n \times n$-undermatrisen oppe til venstre, skyldes at oil-variablene ikke ganges med seg selv. Etter at vi gjemmer det interne mønsteret med den lineære funksjonen s kan vi skrive matrisene $G_i = S \begin{pmatrix} 0 & A_i \\ B_i & C_i \end{pmatrix} S^t$, der S er en inverterbar $2n \times 2n$-matrise.

Definisjon: *Vi definerer oil-(vinegar-)underrommet til å være det lineære underrommet til alle vektorene i K^{2n} der den andre(første) halvdelen består bare av nuller.*

Lemma 5.1. *La E og F være $2n \times 2n$-matriser med en øvre venstre $n \times n$-nullmatrise. Dersom F er inverterbar så er oil-underrommet et invariant underrom til EF^{-1}.*

Bevis: Se [17] □

Vi trenger så et par definisjoner for å komme videre:

Definisjon: *For en inverterbar matrise G_j, definerer vi $G_{ij} = G_i G_j^{-1}$.*

Definisjon: *La O være bildet til oil-underrommet S^{-1}.*

For å finne oil-underrommet bruker vi følgende teorem:

Teorem 5.2. *O er et felles invariant underrom til alle matriser G_{ij}.*

Bevis:

$$G_{ij} = S \begin{pmatrix} 0 & A_i \\ B_i & C_i \end{pmatrix} S^t (S^t)^{-1} \begin{pmatrix} 0 & A_j \\ B_j & C_j \end{pmatrix}^{-1} S^{-1} = S \begin{pmatrix} 0 & A_i \\ B_i & C_i \end{pmatrix} \begin{pmatrix} 0 & A_j \\ B_j & C_j \end{pmatrix}^{-1} S^{-1}$$

De to indre matrisene er på formen E og F fra Lemma 5.1. Derfor er oil-underrommet et invariant underrom til det indre uttrykket og O er et invariant underrom til $G_i G_j^{-1}$. □

Problemet med å finne felles invariante underrom til disse matrisene ser de nærmere på i [17]. Tar vi i bruk algoritmene der, finner vi O. Vi kan da velge et tilfeldig underrom V med dimensjon n slik at $V + O = K^{2n}$, og de gir en ekvivalent separasjon av oil og vinegar. Når vi først får splittet dette opp tar vi tilbake de lineære leddene som vi overså. Vi velger tilfeldige verdier for vinegar-variablene og sitter igjen med en mengde av n lineære ligninger med n oil-variable

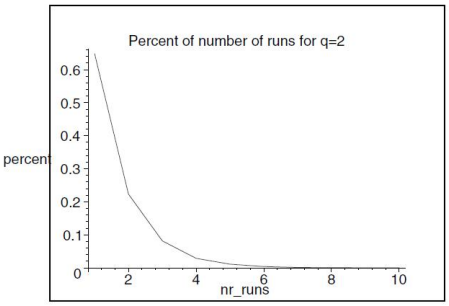
Vi legger merke til at Lemma 5.1 ikke lenger gjelder når $v > n$. Oil-underrommet avbildes fremdeles inn i vinegar-underrommet av E og F, men F^{-1} sender ikke nødvendigvis bildet til E på oil-underrommet tilbake til oil-underrommet og dette er hvorfor kryptanalysen ikke lenger fungerer for den ubalanserte versjonen av dette signaturskjemaet.

5.6.2 Angrep basert på Gröbner-baser

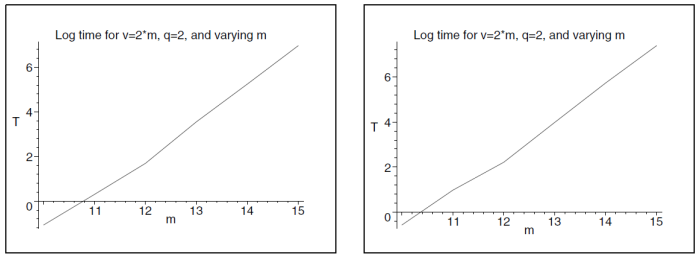
Artikkelen til Courtois, Daum og Felke [6] forklarer en måte å angripe et annet multivariabelt kryptosystem (Hidden Field Equations) ved hjelp av algoritmer for å finne Gröbner-baser. Angrepet fungerer når $m < n$, det vil si at vi har færre ligninger enn variable. Ideen er å legge til $n - m$ ligninger og på denne måten redusere antall variable til m. På den andre siden vil et system med n variable og m ligninger ha q^{n-m} løsninger som forventningsverdi. Ved å legge til $n - m$ lineære ligninger vil man derfor få én løsning i gjennomsnitt. Gjentar vi dette eksperimentet noen ganger vil vi finne minst en løsning. I eksperimentene som Braeken, Wolf og Preneel utførte i [3] fikserte de i stedet $n - m$ variable til en tilfeldig verdi fra kroppen de jobbet med. Ser vi nærmere på dette ser vi at de to fremgangsmåtene er ekvivalente da transformasjonen S allerede gir et tilfeldig system av ligninger av grad 1.

Forfatterne begynte med å beregne gjennomsnittet av antall forsøk nødvendig i en rekke eksperimenter der n tok verdier fra 10 til 24 og v gikk fra 1 til $n-1$. Det viste seg at i over 60% av gangene fikk de en løsning ved den første fikseringen av variable og etter det konvergerte antall nødvendige forsøk raskt mot null.

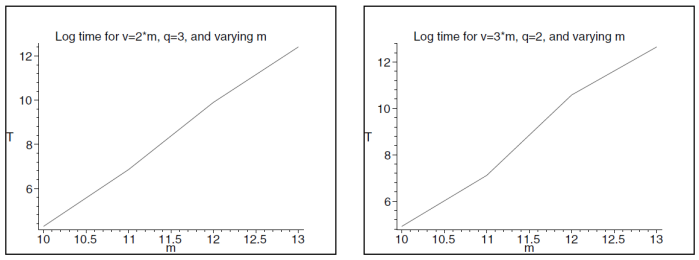
Videre undersøkte de tidskompleksiteten til angrepet for fiksert m og varierende v. Det viste seg at tidskompleksiteten øker eksponentielt med økende v. Siden systemet blir mer tilfeldig med økende v og dermed vanskeligere å løse, kommer ikke dette som noen stor overraskelse. Likevel må vi huske på at antallet løsninger også øker med q^v, altså eksponentielt. Dermed blir sannsynligheten for å finne en av disse løsningene også høyere.



Figur 1: Oversikt over antall nødvendige forsøk, hentet fra [3].



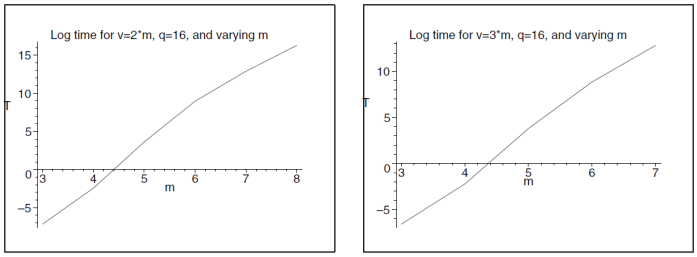
Figur 2: Logaritmisk tid som funksjon av m med henholdsvis $v = 2m$ og $v = 3m$, $q = 2$. Hentet fra [3].



Figur 3: Logaritmisk tid som funksjon av m med henholdsvis $v = 2m$ og $v = 3m$, $q = 3$. Hentet fra [3].

Dernest så de på den logaritmiske tidskompleksiteten (T) mens de endret antallet ligninger m til de to verdiene $v = 2m$ og $v = 3m$ og brukte karakteristikker $q = 2, q = 3$ og $q = 16$.

På bakgrunn av disse forsøkene ble konklusjonen at antallet ligninger bør være over 38 for karakteristikk 2 og over 24 for karakteristikk 3 både for $n \geq 2m$ og $n \geq 3m$. Dette for å få et sikkerhetsnivå på ønsket nivå, det vil si over 2^{64}. I [3] vil de ikke forutsi oppførselen for $q = 16$ siden



Figur 4: Logaritmisk tid som funksjon av m med henholdsvis $v = 2m$ og $v = 3m$, $q = 16$. Hentet fra [3].

tallene ikke konvergerte klart nok og videre forsøk må utføres.

Undersøkelsene her ble utført med MAGMA og brukte implementasjonen til Faugeres algoritme F_4, en algoritme vi skal se nærmere på etter hvert.

5.6.3 Angrep basert på affine underrom

Dette angrepet er en utvidelse av Youssef og Gongs angrep [27] mot det beslektede kryptosystemet Imai-Matsumato Scheme B. Tanken er at et kryptosystem kan bli approksimert av flere affine ligninger. Det opprinnelige angrepet tok bare hensyn til partallskarakteristikker, men dette angrepet er generalisert til odde karakteristikker også.

Angrepet samler flere punkter som tilhører det samme affine underrommet W. Siden vi har w punkter $x_1, \ldots, x_w \in \mathbb{F}^n$ der UOV er affin så kan vi bruke en funksjon $F(x) = Ax + b$ til å beskrive outputen til UOV. For å initiere angrepet beregner vi først den tilhørende $y_i = UOV(x_i)$ for $1 \leq i \leq w$ og $y_i \in \mathbb{F}^m$. Denne kunnskapen gjør at vi kan bestemme for enhver y_i om den tilhører underrommet W og, dersom den gjør det, beregne en vektor $a \in \mathbb{F}^w$ med $y' = \sum_{i=1}^{w} a_i y_i$. Siden underrommet W er affint så kan vi bestemme den korresponderende $x' \in \mathbb{F}^n$ som $\sum_{i=1}^{w} a_i x_i$. Videre skal vi se på et par ulike metoder for å beregne punktene x_i, det vil si å beregne ett eller flere underrom W.

For UOV eksisterer det omtrent q^v disjunkte underrom med dimensjon $o = m$ der UOV er affin. Dersom vi kan finne $(o + 1)$ lineært uavhengige punkter til det samme underrommet, så har vi knekket systemet i dette underrommet. Dersom vi finner færre punkter, for eksempel w, har vi dekket minst q^w punkter i det tilhørende underrommet W. Gjentar vi søket for $(o + 1)$ punkter q^v ganger knekker vi hele systemet. Legg merke til at dersom vårt eneste mål er å forfalske en signatur til en gitt melding $y \in \mathbb{F}^m$ trenger vi bare å kjenne det ene underrommet W der $y \in W$. Vi trenger derfor ikke å kjenne alle q^v underrommene, men bare et lite antall for, med høy sannsynlighet, å kunne forfalske enhver signatur $x \in \mathbb{F}^n$ for en gitt beskjed $y \in \mathbb{F}^m$. Vi husker at $(x, y) \in \mathbb{F}^n \times \mathbb{F}^m$ er definert som et signatur/beskjed-par. Derfor blir oppgaven å beregne $x \in \mathbb{F}^n$ for en gitt $y \in \mathbb{F}^m$.

For å kunne søke etter punkter i samme underrom bruker vi følgende observasjon: dersom de tre punktene $R_1, R_2, R_3 \in \mathbb{F}^n$ er i det samme affine underrommet med tanke på UOV, må følgende krav være oppfylt:

$$UOV(R_1) - UOV(R_2) - UOV(R_3) + UOV(-R_1 + R_2 + R_3) = 0. \tag{1}$$

Vi sjekker validiteten til ligningen over ved å beregne

$$AR_1 + v - AR_2 - v - AR_3 - v + A(-R_1 + R_2 + R_3) + v = A(R_1 - R_1 + R_2 - R_2 + R_3 - R_3).$$

Ved å bruke den egenskapen kan vi bestemme punktene i det samme affine underrommet ved å repetere den heuristiske algoritmen under flere ganger. Den tilsvarende algoritmen for partallig karakteristikk er beskrevet i [27].

input: punkt R_1, offentlig nøkkel P til UOV
output: Et punktpar R_1, R_2 som tilhører samme affine underrom
repeat
  $pass \leftarrow 0$
  $trials \leftarrow 0$
  $R_2 \leftarrow Random(\mathbb{F}^n)$
  $\delta_x \leftarrow -R_1 + R_2$
  repeat
    $trials \leftarrow trials + 1$
    $R_3 \leftarrow Random(\mathbb{F}^n)$
    $R_4 \leftarrow \delta_x + R_3$
    $\delta_y \leftarrow UOV(R_1) - UOV(R_2) - UOV(R_3) + UOV(R_4)$
    if $\delta_y = 0$ **then**
      $pass \leftarrow pass + 1$
    end if
  until $pass > grense || trials > q^v \cdot grense$
until $pass > grense || trials > q^v \cdot grense$
return R_1, R_2

Gjentar vi denne algoritmen nok ganger for et fiksert punkt R_1 får vi $(o+1)$ lineært uavhengige punkter i ett affint underrom. Kjøretiden til algoritmen vil være $O(q^{2v})$ i samsvar med sannsynligheten at R_1, R_2 og R_3 ligger i samme affine underrom.

Dette angrepet kan forbedres ved å bruke relasjonen

$$UOV(R_1) + UOV(R_2) - UOV(R_1 + R_2) = b \tag{2}$$

for en fiksert $b \in \mathbb{F}^m$. I det vi finner et trippel $(R_1, R_2, R_3) \in (\mathbb{F}^n)^3$ med punkter som alle gir $\delta_y = 0$ i parfinningsalgoritmen, så sjekker vi om alle gir den samme konstanten b i (2). Dersom de gjør det vet vi at sannsynligheten for at alle tre punktene er i samme underrom er q^{-2m}. Når vi har gjort dette kan vi benytte oss av en annen algoritme. I stedet for å se på tripler så ser vi nå på par. Dersom paret (R_1, R') gir samme konstant b, har vi funnet en ny kandidat til et punkt i underrommet til R_1. Bruker vi så de andre punktene vi har funnet så langt, kan vi øke sannsynligheten for at R' er med i underrommet med q^{-m} med hvert punkt vi sjekker. Algoritmen kan oppsummeres slik:

1. Finn et trippel $(R_1, R_2, R_3) \in (\mathbb{F}^n)^3$ som tilfredsstiller (1).
2. Bruk dette trippelet og (2) til å finne verdien til $b \in \mathbb{F}^m$.
3. Bruk (2) til å finne flere punkter $R' \in \mathbb{F}^n$ i det samme underrommet.
4. I det vi har $(o+1)$ ulike punkter $R_i \in \mathbb{F}^n$, bestem matrisen A ved gaussisk eliminasjon.

Gjennomsnittlig kjøretid til denne algoritmen er $O(q^{2v} + (n-v)q^v)$ siden vi velger punktene R_2 og R_3 uavhengig av punkt R_1 i det første steget og også R' er uavhengig av R_1. Den totale kjøretiden til å finne totalt $(o+1)$ punkter i det samme underrommet blir derfor $O(q^{2v})$ siden $O(oq^v)$ er neglisjerbar i forhold til $O(q^{2v})$.

Vi kan også forbedre kjøretiden til parfinningsalgoritmen over dersom vi kan bruke litt minne og samtidig har $m > v$, det vil si at vi har *nok* ligninger i forhold til dimensjonen v i det affine underrommet vi skal finne. Siden dette åpenbart ikke er tilfelle for UOV, kommer ikke denne oppgaven til å ta for seg dette.

Fordelen til denne type angrep mot UOV er at vi vet den eksakte strukturen til de affine underrommene. I tillegg er alle disse affine underrommene disjunkte, noe de ikke er i andre lignende kryptosystemer. Dermed er også teoretiske beregninger av kjøretid mye lettere for UOV.

5.7 Oppsummering

5.7.1 Fordeler med UOV

En stor fordel med dette signaturskjemaet er at det matematiske problemet som blir brukt er såkalt kvanteresistent. Det vil si at dersom man i fremtiden greier å lage en kvantedatamaskin som kan håndtere nok tilstander til å knekke de vanlige signaturskjemaene som RSA og ElGamal, så vil ikke dette påvirke sikkerheten til UOV, da det foreløpig ikke finnes noen algoritme som gir en kvantedatamaskin en stor fordel sammenlignet med dagens datamaskiner i å løse multivariate ligningssystem.

I tillegg så ligger det en fordel i de indre operasjonene. Signaturene blir laget og validert bare ved hjelp av addisjon og multiplikasjon av 'små' verdier. Dette er spesielt interessant for systemer med veldig lite ressurser, for eksempel smartkort.

5.7.2 Problemer med UOV

Problemet med dette signaturskjemaet er nøkkellengdene. Den offentlige nøkkelen er et helt ligningssystem. De som vil verifisere en signatur trenger alle ligningene i (P) for å beregne y_i-ene og sammenligne disse med meldingen. Den offentlige nøkkelen vil være i størrelsesorden av noen kilobyte.

I tillegg er UOV et ganske nytt signaturskjema. Noen angrep er kjent og unngått i dag, men den har ikke stått imot vedvarende angrep over lang tid på samme måte som for eksempel RSA. Det er ganske sikkert at det vil komme nye angrep i årene fremover og det er uvisst om dette faktisk er sikkert nok. Derfor bør man være litt forsiktig med å bruke UOV i viktige system og selv det nyeste på dette området kan ikke brukes i kommersielle sammenhenger.

5.7.3 Resistans

Vi så at det første angrepet effektivt tok seg av UOV med $o = v$, det vil si like mange oil- som vinegar-variable. Forfatterne bak denne ideen, Kipnis og Shamir, har brukt samme tankegang til å utvide angrepet til tilfeller der $u \geq o$ og har hatt suksessfulle angrep mot spesielle parametervalg i UOV [18]. Så lenge vi unngår disse parametervalgene når vi lager vårt signaturskjema vil vi unngå dette angrepet.

Det siste angrepet vi så nærmere på, angrepet som baserte seg på affine underrom, fungerer veldig bra i tilfeller der v er liten. Siden dette ikke er tilfelle for UOV er dette angrepet foreløpig ikke optimalt.

I tillegg så vi på UOV sin motstandskraft mot Gröbner-basisangrep. Det viser seg at UOV klarer seg ganske bra mot denne typen angrep, selv om vi bruker Courtois-Daum-Felke-strategien. Kjøretiden blir veldig høy så lenge vi passet på at systemet vårt er stort nok. I denne oppgaven faller det da naturlig å se hvordan vi kan forbedre denne kjøretiden.

6 Forbedringer av Buchbergers algoritme

Siden bruken av Gröbner-baser er et lovende konsept på mange måter er det åpenbart at den klassiske Buchberger-algoritmen må forbedres. Selv den beste implementasjonen greier ikke å finne Gröbner-baser for store problemer. Paradoksalt nok er en standard Buchberger-algoritme veldig enkel å beskrive, men det er mye vanskeligere å forstå hvordan vi skal forbedre en skikkelig implementasjon.

Faugére peker på to mulige forbedringer i [10]. Siden 90% av tiden brukes til å beregne nuller så vil det være nyttig å ha et bedre kriterium for å fjerne unødvendige par. Dette kommer vi tilbake til i Faugéres F_5-algoritme. Den andre forbedringen bryr seg mer om hvilken strategi vi skal bruke. I løpet av en Gröbner-basisberegning er det mange valg vi må ta. Det viser seg vanskelig å parallellisere Buchbergers algoritme, da den fremstår som sterkt sekvensielt, så Faugére konsentrerte seg om å lage en bedre reduksjonsalgoritme. I F_4-algoritmen reduserer han flere polynomer over en liste polynomer *samtidig* ved å bruke teknikker fra lineær algebra som samtidig gir oss en god oversikt over hele prosessen.

6.1 Faugéres F_4-algoritme for beregning av Gröbner-baser

Før vi setter i gang for alvor ser vi på notasjonen vi kommer til å bruke i denne delen av oppgaven. Ringen vi jobber i er R og $R[\underline{x}] = R[x_1, \ldots, x_n]$ er polynomringen. Vi antar at R er kommutativ og noethersk. Mengden med alle leddene i disse variable kaller vi for $T(x_1, \ldots, x_n)$ eller bare T. Vi betegner som vanlig sorteringsmetoden vår over T med $<$.

Vi lar $f, g, p \in R[\underline{x}]$ med $p \neq 0$ og lar F være en endelig undermengde av $R[\underline{x}]$. Da sier vi at:

- f reduseres til g modulo p (skriver $f \underset{p}{\rightarrow} g$) dersom $\exists t \in T(f)$, $\exists s \in T$ slik at $s * \mathrm{LT}(p) = t$ og $g = f - \frac{a}{\mathrm{LC}(p)} * s * p$ der a er koeffisienten til t i p.
- f reduseres til g modulo P (skriver $f \underset{P}{\longrightarrow} g$), dersom $f \underset{p}{\rightarrow}$ for en $p \in P$.
- f er redusibel modulo $p(P)$ dersom det finnes en $g \in R[\underline{x}]$ slik at $f \underset{p}{\rightarrow} g (f \underset{P}{\longrightarrow} g)$.
- f er toppredusibel modulo P dersom det finnes en $g \in R[\underline{x}]$ slik at $f \underset{P}{\longrightarrow} g$ og $\mathrm{LT}(g) < \mathrm{LT}(f)$.
- $f \underset{P}{\overset{*}{\longrightarrow}} g$ er den refleksiv-transitive tillukningen av $\underset{P}{\longrightarrow}$.

Definisjon: *Dersom M er en $s \times m$-matrise, er $M_{i,j}$ element j på rad i i M. Dersom $T_m = [t_1, \ldots, t_m]$ er en sortert mengde ledd, la $(\epsilon_i)_{i=1,\ldots,m}$ være en kanonisk basis til R^m, så betrakter vi den lineære mappingen $\phi_{T_m} : V_{T_m} \rightarrow R^m$ (der V_{T_m} er undermodulen til $R[\underline{x}]$ generert av T_m) slik at $\phi_{T_m}(t_i) = \epsilon_i$. Den resiprokale funksjonen kaller vi ψ_{T_m}. Takket være ψ_{T_m} kan vi se på vektorer i R^n som polynomer. Matrisen med disse egenskapene kaller vi (M, T_M).*

Definisjon: *Dersom (M, T_M) er en slik $s \times m$-matrise kan vi konstruere følgende polynommengde:*

$$\mathrm{Rows}(M, T_M) := \{\psi_{T_M}(\mathrm{row}(M, i)) | i = 1, \ldots, s\} \backslash \{0\}$$

der $\mathrm{row}(M, i)$ er rad i i M. (Det vil si et element i R^m.) På den andre siden, dersom l er en liste med polynomer og T_l en sortert leddmengde så kan vi konstruere en $s \times m$-matrise A, der $s = \mathrm{size}(l)$ og $m = \mathrm{size}(T_l)$:

$$A_{i,j} := \mathrm{coeff}(l[i], T_l[j]), \;\; i = 1, \ldots, s, \; j = 1, \ldots, m.$$

Vi kaller $A^{(l,T_l)}$ for matrisen $(A_{l,j})$.

Definisjon: *La M være en $s \times m$-matrise og $\hat{Y} = [Y_1, \ldots, Y_M]$ nye variable. Da er $F = Rows(M, \hat{Y})$ en mengde ligninger slik at vi kan beregne $\tilde{F}$, en redusert Gröbner-basis til F for en leksikografisk sorteringsmetode der $Y_1 > \cdots > Y_m$. Fra denne matrisen kan vi rekonstruere en matrise $\tilde{M} = A^{(\tilde{F}, \hat{Y})}$. Vi kaller $\tilde{M}$ den (unike) radechelonformen til M. Vi sier også at $\tilde{F}$ er en radechelonbasis til F.*

For polynomer har vi en lignende definisjon:

Definisjon: *La F være en endelig undermengde av $k[x_1, \ldots, x_n]$ og $<$ en sorteringsmetode. Vi definerer $T_<(F)$ til å være* $\mathrm{Sort}(\{T(f) | f \in F\}, <)$, *$A := A^{(F, T_<(F))}$ og $\tilde{A}$ radechelonformen til A. Vi sier at $\tilde{F} = \mathrm{Rows}(\tilde{A}, T_<(F))$ er radechelonformen til F med tanke på $<$.*

Elementæregenskapene til radechelonmatriser kan oppsummeres i det følgende teoremet:

Teorem 6.1. *La M være en $s \times m$-matrise og $\hat{Y}$ nye variable, $F = Rows(M, \hat{Y})$, $\tilde{M}$ radechelonformen til M og $\tilde{F} = Rows(\tilde{M}, \hat{Y})$. Vi definerer*

$$\begin{aligned} \tilde{F}^+ &= \{g \in \tilde{F} | LT(g) \notin LT(F)\}, \\ \tilde{F}^- &= \tilde{F} \backslash \tilde{F}^+ \end{aligned}$$

For enhver undermengde F^* av F slik at $size(F^*) = size(LT(F))$ og $LT(F^*) = LT/F)$, er $G = \tilde{F}^+ \cup F^*$ en triangulær basis til R-modulen V_M generert av F. Det vil si at for alle $f \in V_M$ så finnes det $(\lambda_k)_k$ elementer fra R og $(g_k)_k$ elementer fra G slik at $f = \sum_k \lambda_k g_k$, $LT(g_|) = LT(f)$ og $LT(g_k) > LT(g_{k+1})$.

Bevis: Siden de leddende leddene i G er parvis distinkte er G lineært uavhengig. Vi hevder at det også finnes et genererende system for V_M. Vi antar for selvmotsigelse at det finnes en $f \in V_M$ slik at $f \xrightarrow[*]{G} f' \neq 0$. Ved definisjonen av en Gröbner-basis har vi at $f' \xrightarrow[*]{\tilde{F}} 0$. Dermed er f' toppredusibel modulo $\mathrm{LT}(\tilde{F}) = \mathrm{LT}(\tilde{F}^+) \cup \mathrm{LT}(\tilde{F}^-) = \mathrm{LT}(\tilde{F}^+) \cup \mathrm{LT}(F^*) = \mathrm{LT}(G)$, slik at f' er toppredusibel modulo G. Dette er en motsigelse. □

Dette kan automatisk overføres til teoremet for polynomer:

Korollar 6.2. *La F være en endelig undermengde av E og $<$ en sorteringsmetode. Videre er $\tilde{F}$ radechelonformen til F med tanke på $<$. Vi definerer:*

$$\tilde{F}^+ = \{g \in \tilde{F} | LT(g) \notin LT(F)\}$$

For alle undermengder F^* av F slik at $size(F^*) = size(LT(F))$ og $LT(F^*) = LT(F)$, så er $G = \tilde{F}^+ \cup F^*$ en triangulær basis til V_M, R-modulen generert av F. For alle $f \in V_M$ eksisterer det $(\lambda_k)_k$ elementer fra R og $(g_k)_k$ elementer fra G slik at $f = \sum_k \lambda_k g_k$, $LT(g_1) = LT(f)$ og $LT(g_k) > LT(g_{k+1})$.

Nå begynner vi å nærme oss selve algoritmen. Vi har tidligere vært innom Buchbergers algoritme og vi ser der at det er noen valg vi må foreta. Hva man velger har ingenting å si for riktigheten av algoritmen, men er det mulig å velge litt smart slik at vi kan få ned kjøretiden? De to viktigste tingene vi ser på er:

(i) Velge et kritisk par fra listen over kritiske par.

(ii) Velge rekkefølgen på divisorene når vi dividerer et polynom på en liste polynomer.

Det viser seg at de beste valgene for å få ned kjøretiden bare tar hensyn til det ledende leddet til polynomene [15]. Vi betrakter så tilfellet der alle inputpolynomene har samme ledende ledd. Her er alle de kritiske parene like og det er ikke mulig å velge spesifikt. På en måte kan dette problemet fikses på en enkel og overraskende måte. Vi velger rett og slett ikke. Mer presist kan vi si at i stedet for å velge ett kritisk par på hvert sted, velger vi en undermengde av kritiske par samtidig. Dermed utsetter vi de nødvendige valgene til et annet sted i algoritmen, nemlig den delen med lineær algebra.

Definisjon: *Et kritisk par av to polynomer* (f_i, f_j) *er et element i* $T^2 \times R[\underline{x}] \times T \times R[\underline{x}]$, $\mathrm{Par}(f_i, f_j) := (\mathrm{LCM}_{ij}, t_i, f_i, t_j f_j)$ *slik at*

$$\mathrm{LCM}(\mathrm{Par}(f_i, f_j)) = \mathrm{LCM}_{ij} = \mathrm{LT}(t_i f_i) = \mathrm{LT}(t_j f_j) = \mathrm{LCM}(\mathrm{LT}(f_i), \mathrm{LT}(f_j))$$

Definisjon: *Vi sier at graden til det kritiske paret* $p_{i,j} = \mathrm{Par}(f_i, f_j), \deg(p_{i,j})$, *er* $\deg(\mathrm{LCM}_{i,j})$. *Vi definerer to projeksjoner* $\mathrm{Left}(p_{i,j}) := (t_i, f_i)$ *og* $\mathrm{Right}(p_{i,j}) := (t_j, f_j)$. *Dersom* $(t, p) \in T \times R[\underline{x}]$ *så skriver vi* $\mathrm{mult}((t, p))$ *det evaluerte produktet* $t * p$.

Nå har vi det vi trenger for å se nærmere på den grunnleggende versjonen av algoritmen. Alle matrisene i de følgende algoritmene er representasjonen av en polynomliste gjennom mengden av alle leddene deres.

F_4-algoritmen
input: F er et endelig underrom av $R[\underline{x}]$
En funksjon $\mathcal{S}el : Liste(par) \to Liste(par)$ slik at $\mathcal{S}el(l) \neq \emptyset$ hvis $l \neq \emptyset$
output: En endelig undermengde av $R[\underline{x}]$
$G := F, \tilde{F}_0^+ := F$ og $d := 0$
$P := \{Par(f, g) | f, g \in G \text{ der } f \neq g\}$
while $P \neq \emptyset$ **do**
  $d := d + 1$
  $Pd := \mathcal{S}el(P)$
  $P := P \backslash P_d$
  $L_d := Left(P_d) \cup Right(P_d)$
  $\tilde{F}_d^+ := Reduksjon(L_d, G)$
  for $h \in \tilde{F}_d^+$ **do**
    $P := P \cup \{Par(h, g) | g \in G\}$
    $G := G \cup \{h\}$
  end for
end while
return G

Vi må utvide reduksjonen av et polynom modulo en undermengde i $R[\underline{x}]$ til en reduksjon av en undermengde av $R[\underline{x}]$ modulo en annen undermengde i $R[\underline{x}]$:

Reduksjon
input: L, en endelig undermengde av $T \times k[x_1, \ldots, x_n]$
G, en endelig undermengde av $k[x_1, \ldots, x_n]$
output: En endelig undermengde av $R[\underline{x}]$ (Kan være tom)
$F := SymbolskPreprosess(L, G)$

$\tilde{F}$:= Reduksjon til rad-echelonform av F med tanke på sorteringsvalget $<$
$\tilde{F}^+ := \{f \in \tilde{F} | \mathrm{LT}(f) \notin LT(F)\}$
return $\tilde{F}^+$

Til slutt ser vi hvordan *hovedfunksjonen* til algoritmen fungerer, det vil si konstruksjonen av 'matrisen' F. Denne underalgoritmen kan sees på som en vanlig reduksjon av alle aktuelle algoritmer dersom vi erstatter den vanlige aritmetikken med: la $0 \neq x, 0 \neq y \in R$. Da er $x + y = 1, x * y = 1, x * 0 = 0$ og $x + 0 = 1$. Dette er derfor en symbolsk preprosess

SymbolskPreprosess
input: $\begin{cases} \text{En endelig undermengde av } T \times k[x_1, \ldots, x_n], L \\ \text{En endelig undermengde av } k[x_1, \ldots, x_n], G \end{cases}$
output: En endelig undermengde av $R[\underline{x}]$
$F := \{t * f | (t, f) \in L\}$
$Done := \mathrm{LT}(F)$
while $T(F) \neq Done$ **do**
  $m \backslash Done$ (m er et element i $T(F)$)
  $Done := Done \cup \{m\}$
  if m er toppredusibel modulo G **then**
    $m = m' * \mathrm{LT}(f)$ for en $f \in G$ og en $m' \in T$
    $F := F \cup \{m' * f\}$
  end if
end while
return F

Vi kan legge merke til at den symbolske preprosessen er veldig effektiv siden kjøretiden er lineær i forhold til størrelsen til outputen. Dette gjelder så lenge size(G) er mindre enn den endelige størrelsen til $T(F)$, noe som normalt stemmer.

Lemma 6.3. *La G være en endelig undermengde av $R[\underline{x}]$, L være bildet ved multiplikasjon av en endelig undermengde av $T \times G$ og $\tilde{F}^+ = Reduksjon(L, G)$. Da er for alle $h \in \tilde{F}^+$, $LT(h) \notin Id(LT(G))$.*

Bevis: La F være mengden beregnet av algoritmen $SymbolskPreprosess(L, G)$. Vi antar, for motsigelse, at $\forall h \in \tilde{F}^+$ slik at $t = \mathrm{LT}(h) \in \mathrm{Id}(\mathrm{LT}(G))$. Dermed vil $\mathrm{LT}(g)$ dividere t for en viss $g \in G$. Dermed er t i $T(\tilde{F}^+) \subset T(\tilde{F}) \subset T(F)$ og lar seg toppredusere av g. Dermed er $\frac{t}{\mathrm{LT}(g)} * g$ kommet inn i F ved symbolsk preprosess (eller et annet produkt med samme ledende ledd). Det er en motsigelse mot at $\mathrm{LT}(h) \notin \mathrm{LT}(F)$. □

Det neste lemmaet er nyttig for å bevise riktigheten til algoritmen:

Lemma 6.4. *La G være en endelig undermengde av $R[\underline{x}]$, L være bildet ved multiplikasjon av en endelig undermengde av $T \times G$ og $\tilde{F}^+ = Reduksjon(L, G)$. Da er $\tilde{F}^+$ en undermengde av $Id(G)$. I tillegg har vi for alle f i R-modulen generert av L at $f \xrightarrow[G \cup \tilde{F}^+]{*} 0$.*

Bevis: Vi bruker Korollar 6.2 på F, mengden generert av Symbolsk Preprosess(L, G). Åpenbart er F en undermengde av $F \cup \mathrm{Id}(G)$, men det er også åpenbart at L er en undermengde av $\mathrm{Id}(G)$, slik at F er en undermengde av $\mathrm{Id}(G)$. Dermed vil enhver F^* som oppfylles hypotesen i Teorem

6.1 være en undermengde av $\mathrm{Id}(G)$. Dette fullbyrder beviset siden R-modulen generert av L er en undermodul av R-modulen generert av F. □

Teorem 6.5. *Algoritmen F_4 beregner en Gröbner-basis $G \in R[\underline{x}]$ slik at $F \subseteq G$ og $Id(G) = Id(F)$.*

Bevis: Terminering: Vi antar for motsigelse av while-løkken ikke avsluttes. Vi ser at det finnes en synkende sekvens (d_i) med naturlige tall slik at $\tilde{F}^+_{d_i} \neq \emptyset$ for alle i. La oss så si at q_i er et vilkårlig element i $\tilde{F}^+_{d_i}$. La så U_i være $U_{i-1} + \mathrm{Id}(\mathrm{LT}(q_i))$ for $i > 1$ og $U_0 = (0)$. Ved Lemma 6.3 får vi at $U_{i-1} \subsetneq U_i$. Dette er en direkte motsigelse mot at $R[\underline{x}]$ er noethersk.

Korrekthet: Vi har $G = \bigcup_{d \geq 0} \tilde{F}^+_d$. Vi hevder at de følgende er løkke-invarianter av while-løkken: G er en endelig undermengde av $R[\underline{x}]$ slik at $F \subset G \subset Id(F)$, og $S(g_1, g_2) \xrightarrow[G]{*} 0$ for alle $g_1, g_2 \in G$ slik at $\{g_1, g_2\} \notin P$. Den første påstanden er en direkte konsekvens at den første delen av Lemma 6.4. Vi ser så på den andre: Dersom $\{g_1, g_2\} \notin P$, så vil det si at $\mathrm{Par}(g_1, g_2) = (\mathrm{LCM}_{1,2}, t_1, g_1, t_2, g_2)$ har blitt valgt i det forrige steget (for eksempel steg d) av funksjonen $\mathcal{S}el$. Dermed er $t_1 * g_1$ og $t_2 * g_2$ i L_d, så $S(g_1, g_2)$ er et element i R-modulen generert av L_d og dermed, ved Lemma 6.4, $S(g_1, g_2) \xrightarrow[G]{*} 0$. □

Vi kan legge merke til at dersom $\mathrm{size}(\mathcal{S}el(l)) = 1$ for alle $l \neq \emptyset$ så er F_4-algoritmen identisk med Buchbergers algoritme.

Eksempel: Vi ser nærmere på hvorfor vi bare tar hensyn til ett element i F^+_d og ikke hele F^+_d når vi beviser termineringen:

Dersom $x > y > z$ for en leksikografisk sortering, $F = [f_1 = xy^2 + 1, f_2 = xz^2 + 1, f_3 = y^3 + y^2]$ og $\mathcal{S}el = identiteten$, så får vi $P_1 = \{\mathrm{Par}(f_1, f_2), \mathrm{Par}(f_2, f_3), \mathrm{Par}(f_1, f_3)\}$ og $\tilde{F}^+_1 = \{y^2 - z^2, y + 1\}$ slik at $\mathrm{Id}(\mathrm{LT}(\tilde{F}^+_1)) = \{y\}$. Dermed ser vi at i motsetning til for Buchbergers algoritme vil vi ikke alltid få $G' := G \cup \{h\}$ etter hver gjennomgang. Vi har $\mathrm{Id}(\mathrm{LT}(G')) \supsetneq \mathrm{Id}(\mathrm{LT}(G))$.

6.1.1 Buchbergers kriterium. Forbedrede F_4-algoritmer

For å få en effektiv algoritme må vi inkorporere Buchbergers kriterium i algoritmen over. Vi ser ikke nærmere på forbedringer av dette kriteriet her og bruker dermed en vanlig implementasjon som for eksempel Gebauer og Möller sin [14].

Buchbergerkriteriet

Spesifisering: $\begin{cases} (G_{new}, P_{new}) := Update(G_{old}, P_{old}, h) \\ \text{Oppdatering av listen av kritiske par og idealbasis. (Se [2] side 230)} \end{cases}$

input: $\begin{cases} \text{En endelig undermengde } G_{old} \text{ av } R[\underline{x}] \\ \text{En endelig mengde } P_{old} \text{ av kritiske par i } R[\underline{x}] \\ 0 \neq h \in R[\underline{x}] \end{cases}$

output: En endelig undermengde av $R[\underline{x}]$ og en liste med kritiske par.

I den forrige versjonen av algoritmen brukte vi bare *noen* av radene i den reduserte matrisen (mengdene $\tilde{F}^+$) og forkastet de radene som allerede var i den opprinnelige matrisen (mengdene TF). I den nye versjonen av algoritmen beholder vi disse unyttige radene og prøver å erstatte noen produkter $m * f$ fra radene i 'matrisen' F med nye 'ekvivalente' produkter $m' * f'$ der $' > m'$. Dette er oppgaven til funksjonen $Simplify : T \times R[\underline{x}] \times List(Subset(R[\underline{x}])) \to T \times R[\underline{x}]$. Det tredje argumentet til Simplify er listen over alle tidligere beregnede matriser. En komplett beskrivelse av denne funksjonen kommer lenger nede:

Forbedret f_4-algoritme

input: F, en endelig undermengde av $R[\underline{x}]$
En funksjon $\mathcal{S}el : Liste(par) \to Liste(par)$ slik at $\mathcal{S}el(l) \neq \emptyset$ hvis $l \neq \emptyset$
Oppdatering av den delen av Buchbergers algoritme som velger ut hvilke par som skal beregnes ved å bruke kriteriet fra algoritmen på side 230 i [2].

output: En endelig undermengde av $R[\underline{x}]$.

$G := \emptyset$
$P := \emptyset$
$d := 0$
while $F \neq \emptyset$ **do**
  $f := first(F)$
  $F := F \backslash \{f\}$
  $(G, P) := Update(G, P, f)$
end while
while $P \neq \emptyset$ **do**
  $d := d + 1$
  $P_d := \mathcal{S}el(P)$
  $P := P \backslash P_d$
  $L_d := \text{Left}(P_d) \cup \text{Right}(P_d)$
  $(\tilde{F}_d^+, F_d) := Reduksjon(L_d, G, (F_i)_{d=1,\ldots,(d-1)})$
end while
for $h \in \tilde{F}_d^+$ **do**
  $(G, P) := Update(G, P, h)$
end for
return (G)

Den nye reduksjonsfunksjonen er ganske lik med foregående, men den tar inn et nytt argument samt at den nå returnerer selve resultatet av den symbolske preprosessen:

Reduksjon

input: En endelig undermengde av $T \times k[x_1, \ldots, x_n], L$
En endelig undermengde av $k[x_1, \ldots, x_n], G$
$\mathcal{F} = (F_k)_{k=1,\ldots,(d-1)}$, der F_k er en endelig undermengde av $R[\underline{x}]$

output: To endelig undermengder av $R[\underline{x}]$
$F := SymbolskPreprosess(L, G, \mathcal{F})$
$\tilde{F} :=$ Reduksjon til radechelonform av F med tanke på $<$.
$\tilde{F}^+ := \{f \in \tilde{F} | \text{LT}(f) \notin \text{LT}(F)\}$
return $(\tilde{F}^+, F)$

Vi må også oppdatere vår versjon av den symbolske preprosessen:

SymbolskPreprosess

input: En endelig undermengde av $T \times k[x_1, \ldots, x_n], L$
En endelig undermengde av $k[x_1, \ldots, x_n], G$
$\mathcal{F} = (F_k)_{k=1,\ldots,(d-1)}$ der F_k er en endelig undermengde av $k[x_1, \ldots, x_n]$

output: En endelig undermengde av $R[\underline{x}]$

$F := \{\text{mult}(Simplify(m, f, \mathcal{F}|(m, f) \in L\}$
$Done := \text{LT}(F)$
while $T(F) \neq Done$ **do**
  $m \backslash Done$ (m er et element i $T(F)$)
  $Done := Done \cup \{m\}$
  if m er toppredusibel modulo G **then**
    $m = m' * \text{LT}(f)$ for en $f \in G$ og en $m' \in T$
    $F := F \cup \{\text{mult}(Simplify(m', f, \mathcal{F}\}$
  end if
end while
return F

Nå kan vi endelig fokusere på algoritmen vi nevnte lenger oppe, nemlig simplifiseringsalgoritmen.

Simplify
input: $\begin{cases} \text{Et ledd } t \in T \\ \text{Et polynom } f \in k[x_1, \ldots, x_n] \\ \mathcal{F} = (F_k)_{k=1,\ldots,(d-1)} \text{ der } F_k \text{ er en endelig undermengde av } k[x_1, \ldots, x_n] \end{cases}$
output: Et ikke-evaluert produkt, det vil si et element i $T \times R[\underline{x}]$.
for $u \in$ divisorliste til t **do**
  if $\forall j (1 \leq j < d)$ slik at $(u * f) \in F_j$ **then**
    $\tilde{F}_j$ er radechelonformen til F_j med tanke på $<$.
    det finnes en (unik) $p \in \tilde{F}_j^+$ slik at $\text{LT}(p) = \text{LT}(u * f)$
    if $u \neq t$ **then**
      return $Simplify(\frac{t}{u}, p, \mathcal{F})$
    else
      return $(1, p)$
    end if
  end if
end for
return t, f

Algoritmen vår beregner dermed den ønskede Gröbner-basisen.

Bevis: Terminering: Simplify konstruerer en sekvens (t_k, f_k) der $t_0 = t, f_0 = f$ og $t_{k+1} < t_k$ unntatt kanskje på det siste steget. Siden T er noethersk impliserer dette at algoritmen stopper etter et endelig antall steg r_k.

Korrekthet: Den første delen er rett frem siden $\text{LT}(u_k f_k) = \text{LT}(f_{k+1})$ slik at $\text{LT}(T_k f_k) = \text{LT}(\frac{t_k}{u_k} f_{k-1}) = \text{LT}(t_{k+1} f_{k+1})$. Beviset er induksjon på stegnummeret. Så vi antar $r_k = 1, t' = \frac{t}{u}$ og $u * f \in F_j$, $f' \in \tilde{F}_j$ for en j, med $\text{LT}(f') = \text{LT}(u * f)$. Mengden $F = \{uf\}$ kan suppleres med andre elementer i F_j slik at $\text{LT}(F^*) = \text{LT}(F)$ og $\text{size}(F^*) = \text{size}(\text{LT}(F))$. Vi bruker Korollar 6.2 og finner $(\alpha_k) \in R$, $g_k \in F^* \cup (\tilde{F}_j)_+$, slik at $f' = \sum_k \alpha_k g_k$ og $\text{LT}(g_1) = \text{LT}(f')$ og $\text{LT}(f') > \text{LT}(g_k)$ for $k > 2$. Ved konstruksjon av F^* ser vi at $g_1 = u * f$. Dermed er $f' = \alpha_1 uf + r$ med $\text{LT}(r) < \text{LT}(f')$. Da får vi $\alpha_1 \neq 0$ og vi får $tf = \frac{1}{\alpha_1} t' f' - \frac{1}{\alpha_1} t' r$. □

Eksperimentelle observasjoner forteller oss at det *Simplify* gjør i 95% av tilfellene er å returnere et produkt (x_i, p) der x_i er en variabel (og ofte produktet (x_n, p)) [10]. Denne teknikken ligner en

del på en annen Gröbner-basisalgoritme FGLM som er omtalt i [9] og [26]. Vi kommer ikke til å se nærmere på denne algoritmen i denne oppgaven.

6.1.2 Utvelgelsesstrategi

Det sier seg selv at det er viktig med en god utvelgelsesstrategi for å få ned kjøretiden til algoritmen. I vårt tilfelle vil det si funksjonen $\mathcal{S}el$.

Det å beregne Gröbner-baser i tilfeller med gradssortering er ofte det vanskeligste steget i beregningsprosessen. (De andre stegene er eliminering og dekomponering av idealet.) En grunn til dette er at inputen til algoritmen kan være en vilkårlig undermengde av $R[\underline{x}]$ uten noen form for matematisk struktur. Vi vil derfor gi litt struktur til disse polynomene i starten av beregningene våre og da bruker vi konseptet d-Gröbner-baser.

Definisjon: *Dersom G_d er resultatet av Buchbergers algoritme trunkert til grad d, (det vil si at vi forkaster alle kritiske par som har totalgrad $> d$,) kaller vi G_d en (trunkert) d-Gröbner-basis til I.*

Det følgende teoremet gir struktur til denne listen når polynomene våre er homogene

Teorem 6.6. *For homogene polynomer $f_1, \ldots, f_l$ er G_d en Gröbner-basis* opp til grad d, *det vil si:*

(i) $\xrightarrow[G_d]{*}$ *er veldefinert for polynomer f slik at $deg(f) \leq d$.*

(ii) $\forall p \in I$ *slik at* $deg(p) \leq d \Longrightarrow p \xrightarrow[G_d]{*} 0$

(iii) $S(f,g) \xrightarrow[G_d]{*} 0$ *for* $f, g \in G_d$ *slik at* $deg(LCM(LT(f), LT(g))) \leq d$.

I tillegg finnes det en D_0 slik at for alle $d \geq D_0$ så er $G_d = G_{D_0}$ Gröbner-basisen til I.

Bevis: Se [2] □

En effektiv nullstellensatz kan gi et estimat av D_0. Fra et teoretisk synspunkt vil en slik eksplisitt grense på graden redusere problemet med å finne polynomene G_∞ til å løse et system med lineære ligninger. Denne reduksjonen til lineæralgebra i beregningene for å finne Gröbner-baser har lenge blitt brukt for å analysere kjøretiden til Buchbergers algoritme [19].

I praksis fungerer dette ikke like ofte på grunn av:

- Overestimering av D_0.
- Lineærsystemet blir enormt. Matrisen som genereres er ofte mye større en nødvendig.
- Matrisen til lineærsystemet har en generalisert sylvesterstruktur og det å løse et slikt system på en effektiv måte er vanskelig.

Det viser seg at Buchbergers algoritme og F_4-algoritmen gir stegvise måter å løse systemene på. Den nye algoritmen vil beregne G_{d+1} ut fra G_d. Dermed transformerer algoritmen et matematisk objekt (G_d er unik) til et annet matematisk objekt med en sterkere struktur. Buchbergers algoritme er også stegvis siden den beregner ett og ett polynom etter hverandre, men i den nye algoritmen beregner vi en helt ny trunkert basis.

Teorem 6.6 gjelder ikke for ikke-homogene polynomer. En løsning på dette kan være å homogenisere listen med polynomer, men dette er ikke effektivt for store systemer siden vi beregner mye mer

enn vi må. En bedre måte er å betrakte den såkalte *sukkergraden* [15] til hvert polynom. Vi legger til en 'fantom-homogeniseringsvariabel' og sukkergraden, $\deg_S$, til et polynom er graden til polynomet dersom alle beregningene ble utført med den tillagte variabelen.

Definisjon: *For startpolynomene har vi*: $\deg_s(f_i) = \deg(f_i)$, *for alle* $i = 1, \ldots, m$. *Polynomene som dukker opp i beregningene har form som enten* $p+q$ *eller* $t*p$ *der* $t \in T$ *er et ledd. Vi definerer* $\deg_S(p+q) = \max(\deg_S(p), \deg_S(q))$ *og* $\deg_S(m*p) = \deg(m) + \deg_S(p)$. *Vi sier at* $\deg_S(q)$ *er 'sukkeret' til* q.

Definisjon: *Resultatet til Buchbergers algoritme når vi forkaster alle kritiske par med* $\deg_S > d$ *kaller vi* $G_d^{(S)}$.

Ulempen med denne tilnærmingsmåten er at $G_{d+1}^{(S)} \backslash G_d^{(S)}$ inneholder polynomer av varierende grad og nærmere slutten av beregningene får vi at $\deg_s(p) \gg \deg(p)$.

Vi ser nå på et par muligheter for å implementere funksjonen $\mathcal{S}el$. Den enkleste måten å implementere $\mathcal{S}el$ er rett og slett å ta identiteten. I det tilfellet reduserer man alle kritiske par samtidig. Den beste måten de hadde sett nærmere på i [10] er å velge alle de kritiske parene med minimal totalgrad:

$\mathcal{S}el(P)$
input: En liste med kritiske par P.
output: En liste med kritiske par P_d.
$d := \min\{\deg(\text{LCM}(p)), p \in P\}$
$P_d := \{p \in P | \deg(\text{LCM}(p)) = d\}$
return P_D

Dette kaller vi *normalstrategien* til F_4. Dersom inputpolynomene er homogene har vi allerede en Gröbner-basis opp til grad $d-1$ og $\mathcal{S}el$ velger nøyaktig alle de kritiske parene som er nødvendig for å beregne en Gröbner-basis opp til grad d.

6.1.3 Eksempel på f_4-algoritmen

Det er ikke så lett å få fullstendig innsikt i alle detaljer på et lite eksempel, men vi velger likevel i denne delen av oppgaven å se nærmere på det 4-sykliske problemet. Eksempelet er tatt fra [10].

Vi velger grevlex-sortering og normalstrategien.

$$F = [f_4 = abcd - 1, f_3 = abc + abd + acd + bcd, f_2 = ab + bc + ad + cd, f_1 = a + b + c + d]$$

I starten har vi $G = \{f_4\}$ og $P_1 = \{Pair(f_3, f_4)\}$ slik at $L_1 = \{(1, f_3), (b, f_4)\}$. Vi går så inn i $SymbolskPreprosess(L_1, G, \emptyset)$; $F_1 = L_1, Done = \text{LT}(F_1) = \{ab\}$ og $T(F_1) = \{ad, ab, b^2, bc, bd, cd\}$. Vi velger oss et element i $T(F_1) \backslash Done$, for eksempel as, men ad er toppredusibel av G. Vi har $Done = \{ab, ad\}$, $F_1 = F_1 \cup \{df_4\}$ og $T(F_1) = T(F_1) \cup \{d^2\}$. Siden de andre elementene i $T(F_1)$ ikke er toppredusible av G vil $SymbolskPreprosess$ returnere $\{f_3, bf_4, df_4\}$, som på matriseform blir:

$$A_1 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Radechelonformen til A_1 blir da

$$\tilde{A}_1 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 2 & 0 & 1 & 0 \end{bmatrix}$$

det vil si at $\tilde{F}_1 = \{f_5 = ad + bd + cd + d^2, f_6 = ab + bc - bd - d^2, f_7 = b^2 + 2bd + d^2\}$ og siden $ab, ad \in \mathrm{LT}(F_1)$ får vi $\tilde{F}_{1+} = \{f_7\}$ og nå $G = \{f_4, f_7\}$.

Neste steget blir å se på $P_2 = \{\mathrm{Par}(f_2, f_4)\}$, dermed blir $L_2 = \{(1, f_2), (bc, f_4)\}$ og $\mathcal{F} = \{F_1\}$. I $SymbolskPreprosess$ prøver vi først å simplifisere $(1, f_2)$ og (bc, f_4) med $\mathcal{F}$. Vi ser at $bf_4 \in F_1$ og at f_6 er det unike polynomet i $\tilde{F}_1$ slik at $\mathrm{LT}(f_6) = \mathrm{LT}(bf_4) = ab$. Dermed blir $Simplify(bc, f_4, \mathcal{F}) = (c, f_6)$. Nå blir $F_2 = \{f_2, cf_6\}$ og $T(F_2) = \{abc, bc^2, abd, acd, bcd, cd^2\}$. Vi velger abd som er reduserbart med bdf_4, men igjen kan vi erstatte dette produktet med bf_5. Etter flere steg finner vi ut at $F_2 = \{cf_5, df_7, bf_5, f_2, cf_6\}$

$$A_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

Dermed får vi også at

$$\tilde{A}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & -1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & -1 & 0 \end{bmatrix}.$$

Her ser vi at $\tilde{F}_2 = \{f_9 = acd+bcd+c^2d+cd^2, f_{10} = b^2d+2bd+d^3, f_{11} = abd+bcd-bd^2-d^3, f_{12} = abc - bcd - c^2d + bd^2 - cd^2 + d^3, f_{13} = bc^2 + c^2d - bd^2 - d^3\}$ og $G = \{f_4, f_7, f_{13}\}$.

I det neste steget har vi $L_3 = \{(1, f_1), (bcd, f_4), (c^2, f_7), (b, f_{13})\}$ og vi bruker Simplify-funksjonen rekursivt: $Simplify(bcd, f_4) = Simplify(cd, f_6) = Simplify(d, f_{12}) = (d, f_{12})$. Dermed har vi $F_3 = \{f_1, df_{12}, c^2f_7, bf_{13}\}$. Legg merke til at c^2f_{17} ikke kan forenkles, men veldig ofte har vi bare et polynom ganget med en variabel. Etter flere steg i $SymbolskPreprosess$ finner vi $F_3 = \{f_1, df_{12}, c^2f_7, bf_{13}, df_{13}, df_{10}\}$ og $\tilde{F}_3 = \{f_{15} = c^2b^2 - c^2d^2 + 2bd^3 + 2d^4, f_{16} = abcd - 1, f_{17} = -bcd^2 - c^2d^2 + bd^3 - cd^3 + d^4 + 1, f_{18} = c^2bd + c^2d^2 - bd^3 - d^4, f_{19} = b^2d^2 + 2bd^3 + d^4\}$.

Vi ser at rangen til F_3 bare er 5. Det betyr at det er en redusering til null.

6.1.4 Oppsummering av F_4

Vi har nå omformulert utfordringene i Buchbergers algoritme til lineær algebra og dermed blir neste problemstilling å effektivt løse lineære algebraiske systemer. Dette er lettere siden vi allerede har konstruert matrisen A (antallet rader i A er litt overestimert av symbolsk prediksjon) og vi kan velge å begynne radreduksjonen med en rad fremfor en annen med 'god grunn'. Dersom vi har heltallskoeffisienter har vi en stor fordel siden vi kan bruke iterative algoritmer på hele matrisen. Det er riktignok en del negative aspekter med dette også. Matrisen A er singulær og som regel enorm. Vi kommer ikke til å gå dypere inn i disse problemstillingene i denne oppgaven og henviser derfor til [10].

6.2 Faugéres F_5-algoritme

Vi har akkurat sett på forbedringene av Buchbergers algoritme som tok for seg ulike strategier og hvordan vi skal velge underveis i algoritmen. Det vi nå skal se nærmere på er hvordan vi kan fjerne unyttige beregninger. Siden cirka 90% av tiden brukes til å beregne null så er det et stort potensial i å finne et bedre kriterium for å fjerne unyttige kritiske par.

I [19] finner vi sammenhengen mellom beregningen av en Gröbner-basis til en polynommengde $F = [f_1, \ldots, f_m]$ og lineæralgebra: Buchbergers algoritme kan betraktes som en triangulering av en undermatrise til sylvestermatrisen. Redusering av et polynom til null kan forstås som en lineær avhengighet blant radene til matrisen. Siden hver rad i matrisen er et produkt $t \times f$ der t er et ledd og $f \in F$, så er en lineær avhengighet $\sum \lambda t f = 0$ eller $\sum_{i=1}^{m} g_i f_i = 0$. Med andre ord er $(g_1, \ldots, g_m)$ en syzygy.

Definisjon: *Et element $s \in R[\underline{X}]^m$ kalles en syzygy med tanke på en sekvens $F = (f_1, \ldots, f_m) \in R[\underline{X}]$ dersom $\sum_{i=1}^{m} s_i f_i = 0$. Dersom $m \geq 2$ får vi for hvert par f_i, f_j med $1 \leq i \leq j \leq m$ den trivielle relasjonen $f_i f_j - f_j f_i = 0$. Dette gir opphav til de prinsipale syzygyene på formen $\pi_{i,j} = f_j e_i - f_j e_j$ der e_k er den k-te standardbasisvektoren i $R[\underline{X}]$.*

Flere artikler har tatt for seg dette problemet. Buchberger foreslår et par kriterier i [4], Gebauer og Möller foreslår en metode i [13], mens Mora, Möller og Traverso foreslår i [21] å beregne en Gröbner-basis og en basis til modulen av syzygyer samtidig. Et kritisk par blir ikke vurdert dersom den tilsvarende syzygyen er en lineærkombinasjon av elementer i den foreløpige basisen til syzygy-modulen.

Alle disse forslagene har det til felles at de implisitt eller eksplisitt bruker de trivielle syzygyene $f_i f_j = f_j f_i$. En annen ting er at alle algoritmene ligner veldig på Buchbergers algoritme med det unntak at enkelte reduksjoner unngås. Effektiviteten til disse algoritmene er ikke tilfredsstillende nok, verken i teori eller praksis, fordi mange unyttige kritiske par ikke fjernes.

Strategien til Faugére i F_5-algoritmen er å bare ta hensyn til de trivielle syzygyene der $f_i f_j - f_j f_i = 0$, men å ikke beregne modulen til syzygyene. Dette impliserer to store forskjeller til den vanlige Buchberger-algoritmen og F_4-algoritmen. Først trenger vi å beregne *alle* Gröbner-basisene til de følgende idealene: $(f_m), (f_{m-1}, f_m), \ldots, (f_1, \ldots, f_m)$. Den andre forskjellen er at enkelte reduksjoner ikke er tillatt og dette medfører at en reduksjon av et polynom med en liste polynomer kan bli *flere* polynomer.

En konsekvens av at vi begrenser oss til de trivielle syzygyene er at worst case-scenariet vårt ikke unngår alle unyttige par. Dersom vi for eksempel har det samme polynomet to ganger i de opprinnelige ligningene vil det være en reduksjon til null. Derimot vil vi se at dersom inputen vår er regulær så vil vi ikke ha noen reduksjon til null. Vi legger også merke til at den teoretiske worst case-kjøretiden til F_5 ikke er bedre enn andre algoritmer, men i praksis er den raskere enn alle tidligere implementerte algoritmer [11].

6.2.1 Ideen bak F_5-algoritmen

Vi betrakter de følgende systemer av grad 2 med tre variable x, y, z avhengig av parameteren $b \in \{0, 1\}$:

$$\mathcal{S}_b \begin{cases} f_3 = x^2 + 18xy + 19y^2 + 8xz + 5yz + 7z^2 \\ f_2 = 3x^2 + (7+b)xy + 22xz + 11yz + 22z^2 + 8y^2 \\ f_1 = 6x^2 + 12xy + 4y^2 + 14xz + 9yz + 7z^2 \end{cases}$$

Vi vil nå beregne en Gröbner-basis til f_1, f_2, f_3 modulo 23 for en totalgradssortering med $x > y > z$. Dette kan gjøres med Buchbergers algoritme (inkludert Buchbergers kriterium). Det er 5 unyttige par og 5 nyttige. Først antar vi at $b = 0$. For å beregne Gröbner-basisen fortsetter vi grad for grad. For grad 2 er det ikke noe annet valg enn å konstruere matrisen

$$A_2 = \begin{array}{c} \\ f_3 \\ f_2 \\ f_1 \end{array} \begin{array}{c} \begin{array}{cccccc} x^2 & xy & y^2 & xz & yz & z^2 \end{array} \\ \begin{pmatrix} 1 & 18 & 19 & 8 & 5 & 7 \\ 3 & 7 & 8 & 22 & 11 & 22 \\ 6 & 12 & 4 & 14 & 9 & 7 \end{pmatrix} \end{array}$$

som videre trianguleres til

$$B_2 = \begin{array}{c} \\ f_3 \\ f_2 \\ f_1 \end{array} \begin{array}{c} \begin{array}{cccccc} x^2 & xy & y^2 & xz & yz & z^2 \end{array} \\ \begin{pmatrix} 1 & 18 & 19 & 8 & 5 & 7 \\ 0 & 1 & 3 & 2 & 4 & -1 \\ 0 & 0 & 1 & -11 & -3 & -5 \end{pmatrix} \end{array}$$

og dermed har vi fått to 'nye' polynomer i idealet, $f_4 = xy + 4yz + 2xz + 3y^2 - z^2$ og $f_5 = y^2 - 11xz - 3yz - 5z^2$. For grad 3 begynner vi å konstruere matrisen

$$\begin{array}{c} \\ zf_3 \\ yf_3 \\ xf_3 \\ zf_2 \\ yf_2 \\ xf_2 \\ zf_1 \\ yf_1 \\ xf_1 \end{array} \begin{array}{c} \begin{array}{cccccc} x^3 & x^2y & xy^2 & y^3 & x^2z & \dots \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & \dots \\ 0 & 1 & 18 & 19 & 0 & \dots \\ 1 & 18 & 19 & 0 & 8 & \dots \\ 0 & 0 & 0 & 0 & 3 & \dots \\ 0 & 3 & 7 & 8 & 0 & \dots \\ 3 & 7 & 8 & 0 & 22 & \dots \\ 0 & 0 & 0 & 0 & 6 & \dots \\ 0 & 6 & 12 & 4 & 0 & \dots \\ 6 & 12 & 4 & 0 & 14 & \dots \end{pmatrix} \end{array}$$

For å triangulere denne matrisen så ville vi kanskje begynt med å forenkle radene xf_2 og xf_1 med raden xf_3, men dette er tidssløsing siden dette allerede ble gjort i det forrige steget. For eksempel er $f_4 = -f_2 + 3f_3$ slik at $xf_4 = -xf_2 + 3xf_3$. Dette er en viktig ide i Buchbergers algoritme; forsøk å gjenvinne så mye av de tidligere beregningene som mulig. Det er også klart at vi ikke trenger å putte inn både f_1 og f_4 i matrisen siden de er lineært avhengig. Vi lager derfor en matrise med f_4 og f_5 i stedet for f_1 og f_2.

$$\begin{array}{c} \\ zf_3 \\ yf_3 \\ xf_3 \\ zf_4 \\ yf_4 \\ xf_4 \\ zf_5 \\ yf_5 \\ xf_5 \end{array} \begin{array}{c} \begin{array}{cccccccccc} x^3 & x^2y & xy^2 & y^3 & x^2z & xyz & y^2z & xz^2 & yz^2 & z^3 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 18 & 19 & 8 & 5 & 7 \\ 0 & 1 & 18 & 19 & 0 & 8 & 5 & 0 & 7 & 0 \\ 1 & 18 & 19 & 0 & 8 & 5 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 3 & 2 & 4 & 22 \\ 0 & 0 & 1 & 3 & 0 & 2 & 4 & 0 & 22 & 0 \\ 0 & 1 & 3 & 0 & 2 & 4 & 0 & 22 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 12 & 20 & 18 \\ 0 & 0 & 0 & 1 & 0 & 12 & 20 & 0 & 18 & 0 \\ 0 & 0 & 1 & 0 & 12 & 20 & 0 & 18 & 0 & 0 \end{pmatrix} \end{array}$$

som trianguleres til

$$\begin{array}{c} \\ xf_3 \\ yf_3 \\ yf_2 \\ \mathbf{xf_2} \\ zf_3 \\ zf_2 \\ zf_1 \\ \mathbf{yf_1} \\ \mathbf{xf_1} \end{array} \begin{array}{c} \begin{array}{cccccccccc} x^3 & x^2y & xy^2 & y^3 & x^2z & xyz & y^2z & xz^2 & yz^2 & z^3 \end{array} \\ \begin{pmatrix} 1 & 18 & 19 & 0 & 8 & 5 & 0 & 7 & 0 & 0 \\ & 1 & 18 & 19 & 0 & 8 & 5 & 0 & 7 & 0 \\ & & 1 & 3 & 0 & 2 & 4 & 0 & 22 & 0 \\ & & & 1 & 0 & 0 & 8 & 1 & 18 & 15 \\ & & & & 1 & 18 & 19 & 8 & 5 & 7 \\ & & & & & 1 & 3 & 2 & 4 & 22 \\ & & & & & & 1 & 12 & 20 & 18 \\ & & & & & & & 1 & 11 & 13 \\ & & & & & & & & 1 & 18 \end{pmatrix} \end{array}$$

Vi har dermed konstruert 3 nye polynomer som er markert med fet skrift. For eksempel $f_6 = y^3+8y^2x+xz^2+18yz^2+15z^3$ og vi husker at dette polynomet kommer fra raden xf_4 eller ekvivalent xf_2. Når vi nå kommer til grad 4 kommer vi til en nytt problem: matrisen A_4 med rader

$$x^2f_i,\ xyf_i,\ y^2f_i,\ xzf_i,\ yzf_i,\ x^2f_i,\ i = 1, 2, 3$$

har ikke full rang! (Dette tilsvarer 3 unyttige par i Buchbergers algoritme.) Grunnen er at $f_2f_3 - f_3f_2 = 0$ eller skrevet ut:

$$(3x^2 + (7 + b)xy + 8y^2 + 22xz + 11yz + 22z^2)f_3 - (x^2 - 18xy - 19y^2 - 8xz - 5yz - 7z^2)f_2 = 0$$

Dermed kan vi fjerne raden x^2f_2 fra A_4. Ved å bruke $f_1f_3 - f_3f_1 = 0$ kan vi på samme måte fjerne x^2f_1 fra A_4. Siden vi også har at $f_1f_2 = f_2f_1$ ved vi at det er en unyttig rad til i matrisen A_4. Anta at vi går tilbake til det originale problemet $\mathcal{S}_b$ med $b \in \{0, 1\}$. Vi har da:

$$\begin{aligned} 0 = \quad & (f_2f_1 - f_1f_2) - 3(f_3f_1 - f_1f_3) \\ 0 = \quad & (f_2 - 3f_3)f_1 - f_1f_2 + 3f_1f_3 \\ 0 = \quad & f_4f_1 - f_1f_2 + 3f_1f_3 \\ 0 = \quad & ((1 - b)xy + 4yz + 2xz + 3y^2 - x^2)f_1 - \\ & (6x^2 + \cdots)f_2 + 3(6x^2 + \cdots)f_3 \end{aligned}$$

Vi ser her at vi kan fjerne xyf_1 fra A_4 dersom $b \neq 0$ og yxf_1 dersom $b = 0$. Det vil si at det er umulig å vite uten regning hvilken rad som er unyttig siden dette avhenger av verdien b. På den andre siden kan en kombinasjon av trivielle relasjoner $f_if_j = f_jf_i$ alltid skrives:

$$u(f_2f_1 - f_1f_2) + v(f_3f_1 - f_1f_3) + w(f_2f_3 - f_3f_2)$$

der u, v, w er vilkårlige polynomer. Dette kan skrives om til

$$(uf_2 + vf_3)f_1 - uf_1f_2 - vf_1f_3 + wf_2f_3 - wf_3f_2$$

Dermed er alle (trivielle) relasjoner hf_1 slik at h er i idealet generert av f_2 og f_3. Dermed er det enkelt å fjerne linjer dersom vi allerede har en Gröbner-basis til (f_2, f_3). Mer presist kan vi si at vi alltid kan fjerne radene mf_1 der m er et monom delbart med det ledende leddet til et element i $Id(f_1, \ldots, f_m)$. Dersom G_{Prev} er en allerede beregnet Gröbner-basis for $(f_2, \ldots, f_m)$ og vi vil beregne en Gröbner-basis til $(f_1) + G_{Prev}$ så vil vi konstruere matriser med rader mf_1 slik at m er et monom ikke delbart med det ledende leddet til et element i G_{Prev}.

For å avslutte eksempelet med $b = 0$ og for å bruke de tidligere beregningene bruker vi følgende simplifiseringsregel:

$$xf_2 \to f_6,\ f_2 \to f_4,\ xf_1 \to f_8,\ yf_1 \to f_7,\ f_1 \to f_5.$$

Nå blir radene til matrisen A_4

$$yf_7,\ zf_8,\ zf_7,\ z^2f_5,\ yf_6,\ y^2f_4,\ zf_6,\ yzf_4,\ z^2f_4,\ x^2f_3,\ xyf_3,\ y^2f_3,\ xzf_3,\ yzf_3,\ z^2f_3$$

Vi ser dermed at A_4 nesten er en triangulær matrise med unntak av en 5×5-blokk:

$$\begin{array}{c} \\ z^2f_4 \\ z^2f_5 \\ zf_7 \\ zf_8 \\ yf_7 \end{array} \begin{array}{c} \begin{array}{ccccc} xyz^2 & y^2z^2 & xz^3 & yz^3 & z^4 \end{array} \\ \begin{pmatrix} 1 & 3 & 2 & 4 & 22 \\ 0 & 1 & 12 & 20 & 18 \\ 0 & 0 & 1 & 11 & 13 \\ 0 & 0 & 0 & 1 & 18 \\ 1 & 11 & 0 & 13 & 0 \end{pmatrix} \end{array}$$

Redusering av matrisen gir oss et nytt polynom $f_9 = z^4$. Legg merke til at ingen unyttige par (linjer i matrisen redusert til null) er igjen.

Dette eksempelet viser oss at for å kunne bruke de tidligere beregningene ved lavere grader så trenger vi først å gi et unikt 'navn' eller signatur til hver rad i matrisen. Her så vi for eksempel at radene xf_4 og f_6 egentlig var xf_2. Den andre tingen vi må implementere er simplifiseringsreglene. I resten av oppgaven antar vi at alle polynomene er homogene og at koeffisientene til polynomene ligger i en kropp k og vi betegner polynomringen vår $\mathcal{P} = k[x_1, \ldots, x_n]$. Vi legger også merke til at dersom G er en Gröbner-basis så er $\mathrm{NF}(f, G) = g$ der $f \xrightarrow[G]{*} g$ er normalformen til f med tanke på G.

6.2.2 Signaturen til et polynom

Vi lar $(f_1, \ldots, f_m)$ være et m-tuppel av polynomer (et element i den frie modulen $\mathcal{P}^m$) og I idealet generert av $(f_1, \ldots, f_m)$. Vår oppgave nå er å binde en unik og kanonisk 'signatur' til alle elementene i $T(I)$, altså alle ledende ledd til alle polynomene i idealet.

Vi lar F_i betegne den kanoniske i-ende enhetsvektoren i $\mathcal{P}^m$. Vi betrakter følgende funksjon:

$$v\begin{pmatrix} \mathcal{P}^m & \to & \mathcal{P} \\ g = (g_1, \ldots, g_m) & \mapsto & \sum_{i=1}^m f_i g_i \end{pmatrix}$$

Vi har $v(F_i) = f_i$ og $g = \sum_{i=1}^m g_i F_i$. Et m-tuppel $g = (g_1, \ldots, g_m)$ er en syzygy dersom $v(g) = 0$. De prinsipale syzygyene er $s_{i,j} = f_j F_i - f_i F_j$r. Mengden av alle syzygyer er en modul og skrives Syz. Vi lar $PSyz$ være modulen generert av prinsipale syzygyer. For et generisk (tilfeldig) polynomt system $(f_1, \ldots, f_m)$, $Syz = PSyz$.

Vi utvider den passende sorteringsmetoden vår $<$ på $\mathcal{P}^m$ på følgende vis:

$$\sum_{k=i}^m g_k F_k \prec \sum_{k=j}^m h_k F_k \text{ hvis og bare hvis } \begin{cases} i > j \text{ og } h_j \neq 0 \\ \text{eller} \\ i = j \text{ og } \mathrm{LT}(g_i) < \mathrm{LT}(h_i) \end{cases}$$

Vi ser her at $F_1 \succ F_2 \succ \cdots \succ F_m$. For alle $g \in \mathcal{P}^m$ finnes det en indeks i slik at $g = \sum_{k=i}^m g_k F_k$ med $g_i \neq 0$. Denne i-en betegner vi som *indeksen* til g, index(g). Fir den nye sorteringen $\prec$ får vi

$$\mathrm{LT}(g) = \mathrm{LT}(g_i) F_i.$$

Vi definerer graden til $g = \sum_{i=1}^{m} g_i F_i$, $\deg(g)$ slik:

$$\max\{\deg(g_i) + \deg(f_1) \text{ for } i \in \{1, \ldots, m\}\}$$

Vi lar T_i være $\{tF_i | t \in T\}$ slik at $\mathrm{LT}(g) \in T_i$. Vi får da en mengde signaturer til alle polynomer i idealet I nemlig $T = \cup_{i=1}^{m} T_i$. Dersom $t \in T$ ser vi at $W(t) = \{g \in \mathcal{P}^m | \mathrm{LT}(v(g)) = t\}$ kan inneholde mer en ett element og vi må velge et av dem.

Proposisjon 6.7. *La w være*

$$\begin{pmatrix} T & \to & \mathcal{P}^m \\ t & \mapsto & \min_{\prec} W(t) \end{pmatrix}$$

Dersom $(t_1, t_2) \in T(I)^2$, så er $LT(w(t_1)) \neq LT(w(t_2))$ så lenge $t_1 \neq t_2$.

Bevis: Se [11]. □

Korollar 6.8. *For alle polynomene p i idealet I definerer vi $v_1(p)$ å være $LT(w(LT(p)))$. Dersom p_1 og p_2 er to polynomer med distinkte ledende ledd ($LT(p_1) \neq LT(p_2)$) får vi $v_1(p_1) \neq v_1(p_2)$.*

Bevis: Se [11]. □

I Faugéres F_5-algoritme vil $v_1(p)$ være signaturen til polynomet p. Denne er unik og avhenger ikke av rekkefølgen vi beregner. Vi tar vare på denne informasjonen ved å bruke den interne beskrivelsen av et polynom. Matematisk sett vil denne beskrivelsen bli $R = T \times \mathcal{P}$. Dersom $r = (tF_i.f) \in R$ definerer vi:

(i) $\mathrm{poly}(r) = f \in \mathcal{P}$

(ii) $\mathcal{J}(r) = tF_1 \in T$

(iii) $\mathrm{index}(r) = i \in \mathbb{N}$

Vi kommer til å se når vi utfører algoritmen at egenskapen $\mathcal{J}(r) = v_1(\mathrm{poly}(r))$ bevares. Vi sier at $r \in R$ tillates dersom det finnes $g \in v^{-1}(\mathrm{poly}(r))$ slik at $\mathrm{LT}(g) = \mathcal{J}(r)$. Vi lar $0 \neq \lambda \in k$, $v \in T$, $t = wF_k \in T$ og $r = (uF_i, p) \in R$ og definerer $\lambda r = (uF_i, \lambda p)$, $vt = (vw)F_k$ og $vr = (uvF_i, vp)$. Vi definerer ingen addisjon. Samtidig utvider vi definisjonen til de vanlige operatorene på R:

(i) for $r \in R$, $\mathrm{LT}(r) = \mathrm{LT}(\mathrm{poly}(r))$

(ii) for $r \in R$, $\mathrm{LC}(r) = \mathrm{LC}(\mathrm{poly}(r))$

(iii) for $r \in R$ og $G \subset \mathcal{P}$, $NF(r, G) = (\mathcal{J}(r), NF(\mathrm{poly}(r), G))$.

6.2.3 Nytt kriterium

Definisjon: *La P være en endelig undermengde av R og $r, t \in R$. Dersom* $\mathrm{poly}(r) = \sum_{p \in P} m_p p$, $m_p \in \mathcal{P}$ *sier vi at dette er en t-representasjon av r med tanke på P dersom* $\mathrm{LT}(t) \geq \mathrm{LT}(m_p p)$ *og $\mathcal{J}(r) \geq \mathcal{J}(m_p p)$ for alle $p \in P$. Denne egenskapen fører vi som $r = \mathcal{O}_p(t)$. Vi bruker notasjonen $r = o_p(t)$ dersom det finnes $t' \in R$ slik at $\mathcal{J}(t') \leq \mathcal{J}(t)$ og $\mathrm{LT}(t') < \mathrm{LT}(t)$ slik at $r = \mathcal{O}_p(t')$.*

Definisjon: *Vi sier at $r \in R$ er normalisert dersom $\mathcal{J}(r) = eF_k$ og e ikke er toppreduserbar av $Id(f_{k+1}, \ldots, f_m)$.*

Vi sier at $(u, r) \in T \times R$ er normalisert om ur er normalisert. Vi sier at et par $(r_i, r_j) \in R^2$ er normalisert om $u_j \mathcal{J}(r_j) \prec u_i \mathcal{J}(r_i)$, samt at (u_i, r_i) og (u_j, r_j) er normalisert der $\tau_{i,j} = \mathrm{LCM}(\mathrm{LT}(r_i), \mathrm{LT}(r_j))$, $u_i = \frac{\tau_{i,j}}{\mathrm{LT}(t_i)}$ og $u_j = \frac{\tau_{i,j}}{\mathrm{LT}(r_j)}$.

Teorem 6.9. *La $F = [f_1, \ldots, f_m]$ være en liste med moniske polynomer. La $G = [r_1, \ldots, r_{n_G} \in R^{n_G}$ slik at:*

(i) $F \subset poly(G)$. La $g_i = poly(r_i)$ og $G_1 = [g_1, \ldots, g_{n_G}]$.

(ii) alle r_i-ene tillates og er moniske $(i = 1, \ldots, n_G)$.

(iii) for alle $(i, j) \in \{1, \ldots, n_G\}$ der paret (r_i, r_j) er normalisert så får vi $S(g_i, g_j) = o_G(u_i r_i)$ (eller 0) der

$$u_i = \frac{LCM(LT(g_I), LT(g_j))}{LT(r_i)}$$

Da er G_1 en Gröbner-basis til I.

Bevis: Se [11]. □

6.2.4 Simplifiseringsregler

Vi skal nå se nærmere på hvordan vi skal implementere simplifiseringsreglene, for eksempel $xF_2 \to f_6$ og $F_2 \to f_4$ i eksempelet over. Vi bruker en array som vi kaller *Regel* for å lagre reglene. Hvert element i *Regel* er en liste med elementer i $T \times \mathbb{N}$. I starten har vi ingen regler:

ResetSimplifikasjon
input: m, antallet polynomer
for $i := 1, 2, \ldots, m$ **do**
  $Regel[i] := \emptyset$
end for

LeggTilRegel $(r_k = (tF_i, p) \in R)$
$Regel[i] := concat([[t, k]], Regel[i])$

Følgende prosedyre prøver å forenkle produktet $u \times r_k$

Omskrive $(u \in T, r_k = (tF_i, p) \in R)$
$L := Regel[i] = [[t_1, k_1], \ldots, [t_r, k_r]]$
for $i = 1, \ldots, r$ **do**
  if ut delbar på t_i **then**
    return $(\frac{ut}{t_i}, r_{k_i})$
  end if
end for
return (u, r_k)

Den neste funksjonen returnerer $TRUE$ dersom $u \times r_k$ kan skrives om:

Omskrivbar? $(u \in T, r_k = (tF_i, p) \in R)$
$(v, r_l) := Omskrive(u, r_k)$
return $l \neq k$

Eksempelvis ser vi at om $r_4 = (F_2, f_4)$ og $r_6 = (xF_2, f_6)$ slik det var i forrige eksempel så vil $LeggTilRegel(r_4)$ og $LeggTilRegel(r_6)$ legge til to nye regler, $xF_2 \to f_6$ og $F_2 \to f_4$. Nå vil $Omskrive(xy, r_4)$ returnere (y, r_6) og $Omskrivbar?(y^2, r_4)$ returnere $TRUE$.

6.2.5 Beskrivelse av algoritmen

Vi begynner med å se på hovedløkken. Siden algoritmen er stegvis vil denne iterere på antallet polynomer:

Inkrementalalgoritme F_5
input: $\begin{cases} F = (f_1, \ldots, f_m), \text{ det vil si en liste med homogene polynomer.} \\ \text{Et godkjent sorteringsmetode } < \end{cases}$
$N := m$ (Antallet polynomer $r_1, \ldots, r_N$ som forekommer i algoritmen)
$ResetSimplifikasjon(m)$
$r_m := (F_m, f_m) \in R$
$G_m := [r_m]$
for $i := (m-1), \ldots, 1$ (I den rekkefølgen) **do**
  $G_i := AlgoritmeF_5(i, f_i, G_{i+1})$
end for
return $\text{poly}(G) = [\text{poly}(r) | r \in G_1]$

Definisjon: *Det kritiske paret* $(r_1, r_2) \in R^2$ er:

$$KritPar(r_1, r_2) = (\text{LCM}_{r_1,r_2}, u_1, r_1, u_2, r_2).$$

Dette er et element i $T^2 \times R \times T \times R$ *slik at:*

$$\begin{aligned} \text{LCM}(KritPar(r_1, r_2)) = & \quad \text{LCM}_{r_1,r_2} \\ = & \quad u_1\text{LT}(r_1) = u_2\text{LT}(r_2) \\ = & \quad \text{LCM}(\text{LT}(r_1), \text{LT}(r_2)) \text{ og} \\ \mathcal{J}(u_1 r_1) \succ & \quad \mathcal{J}(u_2 r_2) \end{aligned}$$

Vi sier at graden til et slikt kritisk par er $\deg(\text{LCM}_{r_1,r_2})$.

Vi har nå den grunnleggende versjonen av algoritmen. Vi velger å se på algoritmen på samme måte som vi ser på Buchbergers algoritme, det vil si med polynomer i stedet for lineær algebra. Om vi hadde brukt lineær algebra, som i F_4 ville vi fått raskere kjøretid, men for enkelhetens skyld og for å se bedre forskjell på F_4 og F_5 gjør vi det slik. Den eneste strukturelle forskjellen fra en standard Buchberger-implementering er at reduksjon av ett polynom med tanke på en liste polynomer kan returnere flere polynomer.

Denne algoritmen bruker tre hjelpefunksjoner: definisjonen av $KritPar$ (konstruksjon av kritiske par dersom det nye kriteriet ikke kan brukes), *Spol* (konstruksjon av S-polynomer) og *Reduksjon* (reduksjon av polynomer med tanke på den foreløpige polynomlisten). Først ser vi på selve algoritmen:

Algoritme F_5

input: $\begin{cases} \text{Et heltall } i \text{ og polynomet } f_i \\ G_{i+1} \text{ En endelig undermengde av } R, \text{ slik at poly}(G_{i+1}) \\ \text{ er en Gröbner-basis til } Id(f_{i+1}, \ldots, f_m) \end{cases}$

$r_i := (F_i, f_i) \in R$
$\phi_{i+1} = \text{NF}(., \text{poly}(G_{i+1}))$
$G_i := G_{i+1} \cup \{r_i\}$
$P := \text{Sort}[\text{KritPar}(r_i, r, i, \phi_{i+1}) | r \in G_{i+1}])$ etter grad
while $P \neq \emptyset$ **do**
  $d := \deg(first(P))$
  $P_d := \{p \in P | \deg(p) = d\}$
  $P := P \backslash P_d$
  $F := \text{Spol}(P_d)$
  $R_d := \text{Reduksjon}(F, G_i, i, \phi_{i+1})$
  for $r \in R_d$ **do**
    $P := P \cup \{\text{KritPar}(r, p, i, \phi_{i+1}) | p \in G_i\}$
    $G_i := G_i \cup \{r\}$
  end for
  $P := \text{Sort}(P)$ etter grad
end while
return G_i

Så ser vi på konstrueringen av kritiske par som hjelper oss med implementeringen av det nye kriteriet.

KritPar (r_1, r_2, k, ϕ)

input: $\begin{cases} k, \text{ et heltall} \\ r_1, r_2, \text{ to polynomer i } R \\ \phi, \text{ en normalform} \end{cases}$

$p_i := \text{poly}(r_i)$ for $i = 1, 2$
$t := \text{LCM}(\text{LT}(p_1), \text{LT}(p_2))$
$u_i := \frac{t}{\text{LT}(p_i)}$ for $i = 1, 2$
if $u_1 \mathcal{J}(r_1) \prec u_2 \mathcal{J}(r_2)$ **then**
  Bytt r_1 og r_2
end if
$t_i F_{k_i} := \mathcal{J}(r_i)$ for $i = 1, 2$
if $k_1 > k$ **then**
  return $\emptyset$
end if
if $\phi(u_1 t_1) \neq u_2 t_2$ **then**
  return $\emptyset$
end if
if $k_2 = k$ og $\phi(u_2 t_2) \neq u_2 t_2$ **then**
  return $\emptyset$
end if
return $[t, u_1, r_1, u_2, r_2]$

Spol
input: $P = [p_1, \ldots, p_h]$, en liste med kritiske par
La $p_l = [t_l, t_l, r_{i_l}, v_l, r_{j_l}]$ for $l = 1, \ldots, h$
$F := \emptyset$
for $l \in \{1, 2, \ldots, h\}$ **do**
  if $(!Omskrivbar?(u_l, r_{i_l}))$ og $(!Omskrivbar?(v_l, r_{j_l}))$ **then**
    $N := N + 1$
    $r_N := (u_l \mathcal{J}(r_{i_l}), u_l \text{poly}(r_{i_l}) - v_l \text{poly}(r_{j_l}))$
    $LeggTilRegel(r_N)$
    $F := F \cup \{r_N\}$
  end if
end for
$F := \text{Sort}(F)$ etter økende $\mathcal{J}$
return F

Som sagt tidligere er det en stor forskjell i reduksjonsmetoden her sammenlignet med Buchbergers algoritme. Siden reduksjonen her kan føre til flere polynomer må vi modifisere *Reduksjon*-funksjonen vår. Dette gjør vi ved en hjelpefunksjon *ToppReduks* som utfører et elementært reduksjonssteg. Resultatet av *Toppreduks* er et par (r, F') der $r \in R$ og F' er en liste med polynomer. Dersom $F' = \emptyset$ betyr det at r er irredusibel (eller null). På den andre side, om $F' \neq \emptyset$ er $r = \emptyset$ og vi må kjøre *ToppReduks* på alle elementene i F'.

Reduksjon
input:
- $ToDo$, en endelig liste polynomer
- G, en liste polynomer i R
- k, et heltall
- ϕ, en normalform

$Done := \emptyset$
while $ToDo \neq \emptyset$ **do**
  $h := \min(ToDO)$ med tanke på $mathcalJ$
  $ToDo := ToDo \backslash \{h\}$
  $(h_1, ToDo_1) := ToppReduks(\phi(h), G \cup Done, k, \phi)$
  $Done := Done \cup h_1$
  $ToDo := ToDo \cup ToDo_1$
end while
return $Done$

For å implementere *ToppReduks* trenger vi en funksjon for å teste delbarheten til det ledende leddet til et polynom med tanke på en liste med polynomer. Resultatet blir en divisor eller $\emptyset$ dersom den er (topp-)irredusibel.

Reduserbar?
input:
- r_{i_0}, et polynom i R
- $G = [r_{i_1}, \ldots, g_{i_s}]$, der $g_i \in R$
- k, et heltall
- ϕ, en normalform

$t_j F_{k_j} := \mathcal{J}(r_{i_j}),\ j = 0, 1, \ldots, s$

for $j \in \{1, 2, \ldots, s\}$ **do**
  if Alle de følgende betingelsene er sann **then**
  (a) $u = \frac{\mathrm{LT}(r_{i_0})}{\mathrm{LT}(r_{i_j})} \in T$

  (b) $\phi(ut_j) = ut_j$

  (c) $!Omskrivbar?(u, r_{i_j})$

  (d) $ut_j F_{k_j} \neq t_0 F_{k_0}$
    return r_{i_j}
  end if
end for
return $\emptyset$

Det er lett å gi en forklaring for de fire betingelsene:

(a) Den vanlige divisjonstesten

(b) Tester det nye kriteriet, det vil si om (u, r_{i_j}) er normalisert.

(c) Tester for å se om vi kan bruke tidligere beregninger for å unngå unødvendig kalkulering.

(d) Fjerner identiske rader i matrisen.

ToppReduks
input: $\begin{cases} r_{k_0}, \text{ et polynom i } R \\ G, \text{ en liste polynomer i } R \\ k, \text{ et heltall} \\ \phi, \text{ en normalform} \end{cases}$
if $\mathrm{poly}(r_{k_0}) = 0$ **then**
  Advarsel: Systemet er ikke en regulær sekvens
  return $(\emptyset, \emptyset)$
end if
$r' = Reduserbar?(r_{k_0}, G, k, \phi)$
if $r' = \emptyset$ **then**
  return $(\frac{1}{\mathrm{LT}(r_{k_0})}, \emptyset)$
else
  $r_{k_1} = r'$
  $u = \frac{\mathrm{LT}(r_{k_0})}{\mathrm{LT}(r_{k_1})} \in T$
  if $u\mathcal{J}(r_{k_1}) \prec \mathcal{J}(r_{k_0})$ **then**
    $\mathrm{poly}(r_{k_0}) = \mathrm{poly}(r_{k_0}) - u\mathrm{poly}(r_{k_1})$
    return $(\emptyset, \{r_{k_0}\})$
  else
    $N := N + 1$
    $r_N = (r\mathcal{J}(r_{k_1}), u\mathrm{poly}(r_{k_1}) - \mathrm{poly}(r_{k_0})) \in R$
    $LeggTilRegel(r_N)$
    return $(\emptyset, \{r_N, r_{k_0}\})$
  end if
end if

Beviset for at algoritmen fungerer finnes i [11].

6.2.6 Eksempel på F_5-algoritmen

Vi jobber nå med omvendt leksikografisk totalgradssortering der $x > y > z > t$ og koeffisientene er rasjonale tall. Eksempelet vårt er tatt fra [11].

$$\begin{aligned} f_3 &= x^2y - z^2t \\ f_2 &= xz^2 - y^2t \\ f_1 &= yz^3 - x^2t^2 \end{aligned}$$

Algoritmen beregner påfølgende Gröbner-baser til (f_3), (f_3, f_2) og (f_3, f_2, f_1). Siden de siste beregningene er de vanskeligste kan vi hoppe over disse første stegene. De tilhørende Gröbner-basene er: $G_3 = [r_3]$ og $G_2 = [r_3, r_2, r_4, r_5]$ der $r_3 = (F_3, f_3), r_2 = (F_2, f_2), r_4 = (xyF_2, xy^3t - z^4t)$ og $r_5 = (xyz^2F_2, z^6t - y^5t^2)$. Vi har også at $\phi_2 = NF(., [r_3, r_2, r_4, r_5]), r_1 = (F_1, f_1)$ og dermed $G_1 = G_2 \cup \{r_1\} = [r_3, r_2, r_4, r_5, r_1]$.

Det er fire kritiske par:

$$\begin{aligned} p_7 &= (xyz^3, x, r_1, yz, r_2), \\ p_8 &= (x^2yz^3, x^2, r_1, z^3, r_3), \\ p_9 &= (yz^6t, z^3t, r_1, y, r_5) \text{ og} \\ p_{10} &= (xy^3z^3t, xy^2t, r_1, z^3, r_4) \end{aligned}$$

Vi har også at $\mathcal{J}(p_7), \ldots, \mathcal{J}(p_{10})$ er henholdsvis $xF_1, x^2F_1, z^3F_1, xy^2F_1$ som alle er invariante ved ϕ_2. Vi kjører så videre i kalkuleringene.

$\underline{d = 5}$**:**
Vi kjører $Spol(P_5)$ med $P_5 = [p_7]$ og $P = [p_8.p_9, p_{10})$. Vi får da $r_6 = (xF_1, y^3zt - x^3t^2)$ og $F := [r_6]$. Vi legger til regelen $xF_1 \rightarrow r_6$. Det er åpenbart ingen reduksjon av r_6 av G_1 så det returnerte resultatet er $R_5 = [r_6]$ og $G_1 = [r_3, r_2, r_4, r_5, r_1, r_6]$. Vi oppdaterer listen med kritiske par: $p_{11} = (y^3z^3t, z^2, r_6, y^2t, r_1)$, $p_{12} = (y_3z^6t, z^5, r_6, y^3, r_5)$, $p_{13} = (xy^3zt, x, r_6, z, r_4)$, $p_{14} = (x^2y^3zt, x^2, r_6, y^2zt, r_3)$ og $p_{15} = (xy^3z^2t, xz, r_6, y^3t, r_2)$. Vi sjekker at $\mathcal{J}(z^2r_6) = xz^2F_1$ og $\mathcal{J}(z^5r_6) = xz^5F_1$ er redusible av ϕ_2 slik at parene p_{11} og p_{12} forkastes. Dermed får vi $P = [p_8, p_9, p_{10}.p_{13}, p_{14}, p_{15}]$.

$\underline{d = 6}$**:**
Vi kjører $Spol(P_6)$ med $P_6 = [p_8, p_{13}]$ og $P = [p_9, p_{10}, p_{14}, p_{15}]$. Vi sjekker at $Omskrive(x^2, r_1) = (x, r_6)$ så vi beholder ikke p_8. For det andre paret, p_{13}, har vi $Omskrivbar?(x, r_6) = FALSE$ og $Omskrivbar?(z, r_4) = FALSE$ slik at $r_7 = (x^2F_1, z^5t - x^4t^2)$. Vi legger til en ny regel; $x^2F_1 \rightarrow r_7$. Det er heller ingen reduksjon av r_7 med G_1 så vi ender opp med resultatet $R_6 = [r_7]$ og dermed $G_1 = [r_3, r_2, r_4, r_5, r_1, r_6, r_7]$. Blant de kritiske parene finner vi at $(r_7, r_1), (r_7, r_6), (r_7, r_3)$ og (r_7, r_4) er unyttige. De nye kritiske parene blir da $p_{16} = (z^6t, z, r_7, 1, r_5)$ og $p_{17} = (xz^5t, x, r_7, z^3t, r_2)$.

$\underline{d = 7}$**:**
Vi kjører $Spol(P_7)$ $P_7 = [p_{15}, p_{16}, p_{17}, p_{14}]$ og $P = [p_9, p_{10}]$. Vi sjekker at $Omskrive(xz, r_6) = (z, r_7)$ og dermed beholder vi ikke p_{15}. Vi finner av p_{16} er grei og $r_8 = (x^2zF_1, y^5t^2 - x^4zt^2)$ beregnes. Vi legger til en ny regel; $x^2zF_1 \rightarrow r_8$. Vi ser også at p_{17} funker og beregner $r_9 =$

$(x^3F_1, -x^5t^2 + y^2z^3t^2)$. Vi legger til regelen $x^3F_1 \rightarrow r_9$. Vi ser at $Omskrive(x^2, r_6) = (1, r_9)$ så vi forkaster p_14. Det er nå to S-polynomer for å redusere $F = [r_8, r_9]$. Elementene i F er ikke toppredusibel av G_1 som beskrevet i algoritmen, men det mulig å fullredusere r_9 med $yt^2 \times r_1 : r_9 = (x^3F_1, -x^5t^2 + x^2yt^4)$ og det endelige resultatet blir $r_9 = -\phi_2(r_9) = (x^3F_1, x^5t^2 - z^2t^5)$. Resultatet av *Reduksjon* er $R_7 = [r_9, r_8]$ og vi får $G_1 = [r_3, r_2, r_4, r_5, r_1, r_6, r_7, r_8, r_9]$. De kritiske parene $(r_9, r_1), (r_9, r_6), (r_9, r_7), (r_9, r_2), (r_9, r_3), (r_9, r_4), (r_9, r_5), (r_8, r_1), (r_8, r_6), (r_8, r_7), (r_8, r_9), (r_8, r_2)$ og (r_8, r_5) er unyttige. De nye kritiske parene er $p_{18} = (xy^5t^2, x, r_8, y^2t, r_4)$ og $p_{19} = (x^2y^5t^2, x^2r_8, y^4t^2, r_3)$.

$\underline{d = 8}$**:**
Vi kjører $Spol(P_8)$ der $P_8 = [p_9, p_{10}, p_{18}]$ og $P = [p_{19}]$. Vi finner at p_9 fungerer og $r_{10} = (z^3tF_1, y^6t^2 - x^2z^3t^3)$ beregnes. Vi legger til den nye regelen $z^3tF_1 \rightarrow r_{10}$. Vi sjekker at $Omskrive(xy^2t, r_1) = (y^2t, r_6)$ så vi forkaster p_{10}. På samme måte forkaster vi p_{18} siden $Omskrive(x, r_8) = (z, r_9)$. Siden $r_{10} = \phi_2(r_{10}) = (z^3tF_1, y^6t^2 - xy^2zt^4)$ er fullt redusert, blir resultatet $R_8 = [r_10]$ som gir $G_1 = [r_3, r_2, r_4, r_5, r_1, r_6, r_7, r_8, r_9, r_{10}]$. Alle de nye kritiske parene $(r_{10}, r_i), i = 1, \ldots, 8$ forkastes.

$\underline{d = 9}$**:**
Vi kjører $Spol(P_9)$ med $P_9 = [p_{19}]$ og $P = \emptyset$. Vi ser at $Omskrive(x^2, r_8) = (xz, r_9)$ så vi forkaster p_{19} og får $F = \emptyset$ og $R_9 = \emptyset$. Dermed stopper algoritmen og returnerer G_1.

Vi legger merke til at ingen unyttige par gjenstår. Dersom vi hadde brukt Buchbergers algoritme ville vi hatt 7 unyttige par og 5 nyttige.

6.2.7 Oppsummering av F_5-algoritmen

Den maksimale effektiviteten til F_5-algoritmen forventes når antall ligninger er mindre eller like stort som antallet variabler. På den andre siden forventer man dårlig kjøretid når systemet er overbelastet. For eksempel vil vi få en dårlig prestasjon om man har beregnet en Gröbner-basis med tanke på totalgrad og så kjører F_5-algoritmen på dette resultatet.

Den enkleste optimeringen av F_5-algoritmen vi har sett på vil være å implementere lineæralgebraen til F_4-algoritmen slik at vi får det beste fra begge verdener. Vi har dermed sett hvordan F_5-algoritmen drastisk kan redusere kjøretiden til den vanlige Buchberger-algoritmen.

7 Oppsummering

Vi har i denne oppgaven studert Gröbner-baser og sett nærmere på hvilke fordeler disse gir oss når vi skal utføre beregninger i en multivariabel polynomring. Vi har funnet ut hvordan vi kan finne ut om en polynommengde er en Gröbner-basis samt hvordan vi kan beregne en Gröbner-basis til et ønsket ideal ved hjelp av Buchbergers algoritme.

Vi har sett hvordan Gröbner-baser kan benyttes til å lage kryptosystem, men også sett problemene dette medfører.

Videre så vi på UOV, et signaturskjema over den multivariable polynomringen som ikke tar hensyn til Gröbner-baser i sin oppbygning. Vi så på tre typer angrep der ett av dem tok utgangspunkt i Gröbner-baser. Ingen av dem viste seg særlig effektiv.

Til slutt så vi nærmere på bedre algoritmer for å finne Gröbner-baser. Vi så først på hvordan F_4-algoritmen bruker lineæralgebra og annen valgstrategi for å forbedre Buchbergers algoritme. Deretter så vi på F_5-algoritmen og hvordan denne forbedret Buchbergers algoritme ved å forhindre unødvendige beregninger. Vi har dermed sett at begge disse algoritmene kan gi oss en vesentlig forbedring av kjøretiden sammenlignet med den originale Buchberger-algoritmen.

Referanser

[1] B. Barkee, D. C .Can, J. Ecks, T.Moriarty og R.F.Ree. Why you cannot even hope to use Gröbner bases in Public Key Cryptography: an open letter to a scientist who failed and challenge to those who have not yet failed. J. Symb. Comput., 18(6): Side 497-501, 1994.

[2] T. Becker og V. Weispfenning. Groebner Bases, a Computational Approach to Commutative Algebra. Graduate Texts in Mathematics. Springer-Verlag, 1993.

[3] A. Braeken, C. Wolf og B. Preneel. A study of the security of Unbalanced Oil and Vinegar signature schemes. I The Cryptographer's Track at RSA Conference 2005, volume 3376 of Lecture Notes in Computer Science, 2005. http://eprint.iacr.org/2004/222

[4] B. Buchberger. A Criterion for Detecting Unnecessary Reduction in the Construction of Gröbner Basis. I Proc. EUROSAM 79, volum 72 av Lecture Notes in Computer Science, 1979.

[5] M. Caboara, F. Caruso, C. Traverso. Gröbner Bases for Public Key Cryptography. http://posso.dm.unipi.it/crypto/ Eurocrypt 2008 Rump Session, 15. april 2008.

[6] N. T. Courtois, M. Daum og P. Felke. On the security og HFE, HFEv- and Quartz. I Public Key Cryptography –PKC 2003, volume 2567 of Lecture Notes in Computer Science, side 337-350. http://eprint.iacr.org/2002/138 2003.

[7] David A. Cox, John B. Little og Donal O'Shea. Ideals, Varieties and Algorithms. Springer forlag. ISBN 9783540978473

[8] A. Dickenstein, N. Fitchas, M.Giusti, C.Sessa. The membership problem for unmixed polynomial ideals is solvable in single exponential time. Discrete Applied Mathematics 33, side 73-94, 1991.

[9] J.-C. Faugére, P. Gianni, D. Lazard og T. Mora. Efficient computation of zero dimensional gröbner bases by change og ordering. Journal og Symbolic Computation, 1989.

[10] J.-C. Faugére. A new efficient algorithm for computing gröbner bases (f_4). Journal of Pure and Applied Algebra, 139(1): Side 61-88, Juni 1999.

[11] J.-C. Faugére. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). I Proceedings of the 2002 international symposium on Symbolic and algebraic computation. Side 75-83, juli 2002.

[12] M. Garey, D. Johnson. Computers and Intractability, a Guide to the Theory of NP-Completeness, Freeman, side 251.

[13] R. Gebauer og H.M. Möller. Buchberger's algorithm and staggered linear bases. I Proceedings of the 1986 Symposium on Symbolic and Algebraic Computation, juli 1986.

[14] R. Gebauer og H.M. Möller. On an Installation of Buchberger's Algorithm. Journal of Symbolic Computation, Side: 275-286, October/December 1988.

[15] A. Giovini, T. Mora, G. Niesi, L. Robbiano og C. Traverso. One sugar cube, please, or Selection strategies in the Buchberger Algorithm. I Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation. ISSAC '91, 1991.

[16] M. Giusti. Some effectivity problems in polynomial ideal theory. EUROSAM 84. Springer L.N.C.S 174, side 159-171, 1984.

[17] A. Kipnis, A. Shamir, Cryptanalysis of the Oil and Vinegar Signature Scheme, Proceedings of CRYPTO'98, Springer, volume 1462 i Lecture Notes in Computer Science, side 257-266, 1998.

[18] A. Kipnis, J. Patarin og L. Goubin. Unbalanced Oil and Vinegar signature schemes. I Advances in Cryptology –EUROCRYPT 1999, volume 1592 i Lecture Notes in Computer Science, side 206-222, 1999.

[19] D. Lazard. Gaussian Elimination and Resolution of Systems of Algebraic Equations. I Proc. EUROCAL 83, volum 162 av Lect. Noted in Comp Sci, side 146-157, 1983.

[20] T. Matsumoto og H. Imai. Public quadratic polynomial-tuples for efficiant signature verification and message-encryption. I Advances in Cryptology –EUROCRYPT 1988, volume 330 i Lecture Notes in Computer Science, side 419-545, 1988.

[21] T. Mora, H.M. Möller og C. Traverso. Gröbner Bases Computation Using Sysygies. I ISSAC 92, side 320-328, juli 1992.

[22] J. Patarin. Hidden Field Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of asymmetric algorithms. I Advances in Cryptology –EUROCRYPT 1996, volume 1070 i Lectune Notes in Computer Science, side 33-48, 1996.

[23] J. Patarin. The oil and vinegar signature scheme. Lagt frem ved Dagstuhl Workshop on Cryptography, transparenter, September 1997.

[24] P. W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. Revisert utgave av den opprinnelige utgivelsen til P.W. Shor. Dette er en utvidet versjon av artikkelen som ble trykket i Proceedings of the 35th Annual Symposium on Foundation of Computer Science, Santa Fe, New Mexico, http://arxiv.org/abs/quant-ph/9508027, november 1994,

[25] L.M.K. Vandersypen et al. Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. Nature 414, side 883-887, 2001.

[26] T. Wichmann, Der FGLM-Algorithmus: verallgemeinert und implementiert in SINGULAR. Diplomarbeit im Fachbereich Mathematik, Kaiserslautern 1997.

[27] A. M. Youssef og G. Gong. Cryptanalysis of Imai and Matsumoto scheme B asymmetric cryptosystem. I Progress in Cryptology - INDOCRYPT 2001, volum 2247 Lecture Noted in Computer Science, side 214-222, Springer, 2001.