SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

En experimentell studie av termreduktion med Gröbnerbaser och en alternativ algoritm

av

Filip Ström

2025 - No K32

En experimentell studie av termreduktion med Gröbnerbaser och en alternativ algoritm

Filip Ström

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Samuel Lundqvist

2025

Sammanfattning

Detta arbete syftar till att jämföra två metoder i form av algoritmer för reducering av termer. Reducering beskrivs här som processen att bearbeta ett uttryck tills det antar en normalform. Ett klassiskt exempel på reducering som introduceras tidigt i matematikutbildningar är Gausselemination. Uttrycken som bearbetas i detta arbete kallas termer vilket kan beskrivas som produkten av en eller flertal variabler med eller utan upprepning och en koefficient. Den första av de två algoritmerna som undersöks i detta projekt utför termreducering med en reducerad Gröbnerbas. En Gröbnerbas är en särskild representation av ett ideal, strukturerat på ett sätt som gör det enkelt att avgöra om ett polynom tillhör idealet, och kan skapas av Buchbergers-algoritmen. Vidare, den senare alternativa algoritmen given av [JLN24] som hädanefter kommer att refereras till som den alternativa algoritmen. Alternativa algoritmen fungerar enbart på en specifik form av ideal. Skillnaden mellan dem kan komma att påverka beräkningstiden hos beräkningsprogrammet. Detta kan man jämföra med hjälp av ett Pythonprogram som utför beräkningarna medan den mäter iterationssteg. Detta resulterade i att reducering med en reducerad Gröbnerbas gav till stor del färre iterationssteg än den alternativa-algoritmen med avseende på ideal som slumpmässigt genererades med specifika egenskaper.

Abstract

The purpose of this paper is to compare two methods, in the form of algorithms, for reducing terms. In this context, reduction is described as the process of transforming an expression until it reaches a normal form. A classical example of reduction, commonly introduced early in mathematical education, is Gaussian elimination. The expressions being processed in this study are referred to as terms, which can be described as the product of one or more variables-possibly repeated-multiplied by a coefficient. The first of the two algorithms examined in this project performs term reduction using a reduced Gröbner basis. A Gröbner basis is a special representation of an ideal, structured in such a way that it facilitates determining whether a polynomial belongs to the ideal, and can be computed using Buchberger's algorithm. The second algorithm, referred to henceforth as the alternative algorithm, is presented in [JLN24] and only applies to ideals of a specific form. The difference between the two algorithms may affect the computation time of the reduction process. This can be measured using a Python program that performs the calculations while counting the number of iteration steps. The results showed that reduction using a reduced Gröbner basis generally required fewer iteration steps than the alternative algorithm, for ideals that were randomly generated with specific properties.

1 Förord

I detta arbete har LLM använts som stöd för språkgranskning, strukturering och förklaringar av matematiska begrepp. Overleafs inbyggda GPT-implementation har använts för språkgranskning, medan ChatGPT har nyttjats för matematiska förklaringar. Allt innehåll har granskats och formulerats av författaren.

Innehåll

2 Introduktion

I detta arbete undersöks två olika algoritmer för att reducera termer med avseende på ett ideal. Den första algoritmen bygger på Gröbnerbaser som är ett klassiskt verktyg inom datoralgebra för att hitta en entydig normalform för ett polynom. Den andra är en nyare alternativ algoritm introducerad av [JLN24] som är särskilt anpassad för ideal av en viss struktur.

Målet är att experimentellt jämföra hur dessa två algoritmer beter sig vid reduktion av termer när det gäller antalet iterationssteg. Då Gröbnerbaser kan vara tunga att beräkna, men ger kraftfulla verktyg, är det intressant att se hur en mer specialiserad och direkt metod kan stå sig i jämförelse, särskilt när idealet har en speciell form.

3 Frågeställning

Hur skiljer sig antalet reduktionssteg mellan Gröbnerbasreduktion och en alternativ algoritm vid termreduktion i homogena binomialideal av fullständig skärning? I vilken utsträckning kan idealens struktur eller termens form förklara dessa skillnader?

4 Teoretisk bakgrund

4.1 Polynom och monom

Definition 4.1. ([CLO06, Sid. 1]) Ett **monom** med variablerna $x_1, x_2, \ldots, x_n$ är en produkt av formen

$$x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdots x_n^{\alpha_n}$$

där alla exponenter $\alpha_1, \alpha_2, \ldots, \alpha_n$ är icke-negativa heltal.

Exempel 1. Ett monom kan vara $x_1^2 x_2$ eller $x_1^4 x_3^2$ men även 1 då till exempel $x_1^0 x_2^0 = 1$ men $x_1^{-1} x_2$ och 0 är inte monom.

Definition 4.2. ([CLO06, sid. 497]) En **kommutativ ring** är en mängd R där $\cdot$ och $+$ är två binära operationer som uppfyller

1. $(a+b)+c = a+(b+c)$ och $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ för alla $a, b, c \in R$ (associativ)
2. $a + b = b + a$ och $a \cdot b = b \cdot a$ för alla $a, b \in R$ (Kommutativ)
3. $a \cdot (b + c) = a \cdot b + a \cdot c$ för alla $a, b, c \in R$ (distributiv)
4. Det finns $0, 1 \in R$ så att $a + 0 = a \cdot 1 = a$ för alla $a \in R$ (identiteter)
5. Givet $a \in R$, så finns det ett $b \in R$ så att $a + b = 0$ (Additiv invers)

Definition 4.3. ([Frö97, sid. 5][CLO06, sid. 497]) En **kropp** k är en kommutativ ring där varje element förutom den additiva identiteten 0 har en multiplikativ invers, det vill säga för varje $a \in k$ med $a \neq 0$ finns ett element $a^{-1} \in k$ sådant

att $a \cdot a^{-1} = 1$. Elementen 0 och 1 tillhör k och representerar den additiva respektive multiplikativa identiteten, så att för alla $a \in k$ gäller att $a + 0 = a$ och $a \cdot 1 = a$.

Exempel 2. Exempel på kroppar inkluderar de rationella talen $\mathbb{Q}$, de reella talen $\mathbb{R}$ och de komplexa talen $\mathbb{C}$, alla med vanlig addition och multiplikation. Ett exempel på en ändlig kropp är $\mathbb{Z}_p$, där $\mathbb{Z}_p$ definieras som kvotringen av heltalen modulo p, för ett primtal p. Denna kropp har exakt p element.

Definition 4.4. En **term** är ett monom m med en koefficient $c \in k$ skrivet så att termen $t = c \cdot m$.

Definition 4.5. Ett **polynom** p med variablerna $x_1, x_2, \ldots, x_n$ och koefficienter i en godtycklig kropp k är en ändlig linjär kombination av termer. Alltså kan man skriva p

$$p = \sum_{i=1}^{m} c_i \, x_1^{\alpha_{i1}} x_2^{\alpha_{i2}} \cdots x_n^{\alpha_{in}},$$

där m är ett icke-negativt heltal, $c_i \in k$ och för varje exponent $\alpha_{ij} \in \mathbb{N}$ där $i = 1, 2, \ldots, m$ och $j = 1, 2, \ldots, n$.

Definition 4.6. Totalgraden eller graden d för ett polynom är det största värdet av $\alpha_1 + \alpha_2 + \cdots + \alpha_n$ över alla termer $c \cdot x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$ med $c \neq 0$.

Definition 4.7. Homogena polynom är polynom som har samma totalgrad d i alla termer.

Exempel 3. Ett homogent polynom kan vara $2x_2^2 + 3x_3x_4$ eller $x_1^3 + x_3^1 x_4^2$

Definition 4.8. Ett **binom** är ett polynom som enbart består av två termer, så som

$$b = am_1 + bm_2$$

där m_1 och m_2 är monom och $a, b \in k$. Ytterligare så är $m_1 \neq m_2$ och $a, b \neq 0$.

Exempel 4. Ett exempel på ett homogent binom där antalet variabler $n = 2$ och $d = 1$ är $x_1 - 2x_2$.

Definition 4.9. ([CLO06, sid. 2]) En **polynomring** består av polynom i variablerna $x_1, x_2, \ldots, x_n$, där n är ett icke-negativt heltal och koefficienterna tillhör en kropp k. Denna mängd betecknas

$$k[x_1, x_2, \ldots, x_n]$$

och bildar, med polynomaddition och polynommultiplikation, en kommutativ ring.

Definition 4.10. ([Frö97, sid. 45] [CLO06, sid. 53-56]) En **monomordning** är en ordning $>$ på mängden av monom som uppfyller följande egenskaper:

- För varje par av monom m_1 och m_2 gäller exakt ett av följande $m_1 < m_2$, $m_1 = m_2$ eller $m_1 > m_2$ (total ordning).

- Ordningen är välordnad, det vill säga varje icke-tom mängd av monom har ett minsta element.
- Om $m_1 < m_2$, så gäller även $m + m_1 < m + m_2$ för alla monom m (kompatibilitet med multiplikation).

I detta projekt används främst följande tre typer av ordningar:

1. **Lexikografisk ordning (lex)**: Först jämför exponenterna från vänster till höger. Exempelvis är $x^2y < x^2z$ eftersom $y < z$.
2. **Graderad lexikografisk ordning (grlex)**: Först jämför efter totalgrad. Om graderna är lika används lexikografisk ordning.
3. **Graderad omvänd lexikografisk ordning (grevlex)**: Först jämförs efter totalgrad, och om graderna är lika jämför man de exponenter där skillnaden först uppstår men i omvänd ordning.

Definition 4.11. Ledande monom definieras som $lm(p)$ där $p = c_1m_1 + c_2m_2 + \cdots + c_sm_s$ då $m_1 > m_2 > \cdots > m_s$ enligt en bestämd monomordning. Då är $lm(p) = m_1$ det ledande monomet.

Definition 4.12. En **ledande term** är den termen som innehåller det ledande monomet. Den ledande termens koefficient kallas även för **ledande koefficient**. Funktionen $lt(p)$ där p är ett polynom, får ut den ledande termen. Då $F = \{f_1, f_2, \ldots, f_s\}$ så är

$$lt(F) = \{lt(f_1), lt(f_2), \ldots, lt(f_s)\}.$$

Exempel 5. Givet polynomet $f_1 = 2x_1 + 4x_2$ så är $lm(f_1)$ i lexikografisk ordning x_1 och $lt(f_1) = 2x_1$.

Definition 4.13. Ett **moniskt** polynom är ett polynom p där $lt(p) = lm(p)$. Det vill säga, den ledande termens koefficient är 1.

4.2 Minsta gemensamma multipel

Definition 4.14. ([CLO06, sid. 81]) **Minsta gemensamma multipel** av två monom skrivs $mgm(m_1, m_2)$. Då monomet $m_1 = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdots x_n^{\alpha_n}$ och monomet $m_2 = x_1^{b_1} \cdot x_2^{b_2} \cdots x_n^{b_n}$ så är

$$mgm(m_1, m_2) = x_1^{max(\alpha_1, b_1)} \cdot x_2^{max(\alpha_1, b_2)} \cdots x_n^{max(\alpha_n, b_n)}.$$

Exempel 6. Låt monomen vara $m_1 = x_1x_2$ och $m_2 = x_1^2$ då är $mgm(m_1, m_2) = mgm(x_1x_2, x_1^2) = x_1^2x_2$.

Definition 4.15. ([CLO06, sid. 81]) Låt f_1, f_2 vara polynom som är ordnade, **S-polynomet** av f_1, f_2 är då

$$S(f_1, f_2) = \frac{\text{mgm}(\text{lm}(f_1), \text{lm}(f_2))}{\text{lt}(f_1)} f_1 - \frac{\text{mgm}(\text{lm}(f_1), \text{lm}(f_2))}{\text{lt}(f_2)} f_2.$$

Lägg märke till att de ledande koefficienterna i polynomen kommer att divideras då $lt(f_1) = c_1 m_1$, $lt(f_2) = c_2 m_2$ och $mgm(lm(f_1), lm(f_2)) = m_3$ är ett monom så kommer $\frac{m_3}{c_1 m_2} \cdot f_1$ göra f_1 moniskt då den ledande koefficienten dividerar bort. Detsamma sker för f_2 och då kommer båda ledande termerna ta ut varandra. Detta är en egenskap som gör S-polynom särskilt användbara.

Exempel 7. Låt $f_1 = 3x_1^2$ och $f_2 = 2x_1x_2 + x_3^2$ och $k = \mathbb{Z}_{\not 9}$. För att beräkna $S(f_1, f_2)$ tas det fram de ledande monomen i polynomen enligt graderad lexikografi ordning. Det ledande monomet i f_1 är $lm(f_1) = x_1^2$ och det ledande monomet i f_2 är $lm(f_2) = x_1x_2$. Minsta gemensamma multipel av dessa är $mgm(lm(f_1), lm(f_2)) = x_1^2x_2$.

Nu delar vi $x_1^2x_2$ med respektive ledande term, $lt(f_1) = 3x_1^2$, $lt(f_2) = 2x_1x_2$. Från f_1 får vi kvoten $5x_2$, och för f_2 får vi kvoten $4x_1$.

Vi multiplicerar kvoterna med respektive polynom

$$5x_2 \cdot f_1 = 5x_2 \cdot 3x_1^2 = x_1^2x_2$$

$$4x_1 \cdot f_2 = 4x_1 \cdot (2x_1x_2 + x_3^2) = x_1^2x_2 + 4x_1x_3^2$$

För att sedan subtrahera uttrycken så att

$$x_1^2x_2 - (x_1^2x_2 + 4x_1x_3^2) = -4x_1x_3^2$$

Resultatet blir en term, eftersom de ledande termerna tar ut varandra så att endast en term blir kvar.

4.3 Polynomreducering

Definition 4.16. ([CLO06, Sid 62]) Låt f vara ett polynom och $F = \{f_1, f_2, \ldots, f_l\}$ vara en mängd polynom då ges reduceringsalgoritmen enligt följande

Indata: $f, F = (f_1, f_2, \ldots, f_l)$
Utdata: Resten r som blir över
$p \leftarrow f$;
upprepa
  $delbar \leftarrow falskt$;
  för *varje* f *i* F **gör**
    om *lt(f)/lt(p)* **så**
      $p \leftarrow p - (\mathrm{lt}(p)/\mathrm{lt}(f)) \cdot f$;
      $delbar \leftarrow sant$;
      bryt loop
    slut om
  slutför
  om $delbar = falskt$ **så**
    $r \leftarrow r + \mathrm{lt}(p)$;
    $p \leftarrow p - \mathrm{lt}(p)$;
  slut om
tills $p = 0$;

Algorithm 1: R(f)

och benämns som **reducering av** p **med** F. Att reduktionen är en ändlig algoritm visas i [CLO06, Sid 62]. Notera att algoritmen söker efter ett $f_i \in F$ vars ledande term delar ledande termen i p, och använder det första sådana som hittas. Eftersom ingen fast ordning är specificerad för mängden F, kan olika körningar av algoritmen välja olika f_i vid delbarhet, vilket kan påverka resultatet. Därför är reduktionen inte entydig i allmänhet.

Exempel 8. Låt $p = x_1^2 - x_1x_2$, och låt $F = \{f_1, f_2\}$, där

$$f_1 = x_1^2 - x_2^2, \quad f_2 = x_1x_2 + x_2^2.$$

Vi reducerar p med mängden F enligt algoritm $R(p, F)$, där ordningen på polynomen är fixerad så att f_1 behandlas före f_2.

I första steget delar $\mathrm{lt}(f_1) = x_1^2$ den ledande termen $\mathrm{lt}(p) = x_1^2$, och vi får

$$p = p - \frac{x_1^2}{x_1^2} \cdot f_1 = (x_1^2 - x_1x_2) - (x_1^2 - x_2^2) = -x_1x_2 + x_2^2.$$

Därefter delar $\mathrm{lt}(f_2) = x_1x_2$ den nya ledande termen $\mathrm{lt}(p) = x_1x_2$, vilket ger

$$p = (-x_1x_2 + x_2^2) - \frac{-x_1x_2}{x_1x_2} \cdot f_2 = (-x_1x_2 + x_2^2) + (x_1x_2 + x_2^2) = 2x_2^2.$$

Resultatet efter reduktion är alltså $2x_2^2$.

4.4 Ideal

Definition 4.17. ([CLO06, sid. 29]) En delmängd $I \subset k[x_1, x_2, ..., x_n]$ är ett **ideal** om dessa följande egenskaper finns

1. $0 \in I$.
2. Om $f, g \in I$, då är $f + g \in I$ (Slutet under multiplikation).
3. Om $f \in I$ och $h \in k[x_1, x_2, \ldots, x_n]$, då är $h \cdot f \in I$ (Slutet under addition).

Definition 4.18. ([CLO06, sid. 29]) Låt $f_1, f_2, \ldots, f_s$ vara polynom i $k[x_1, x_2, \ldots, x_n]$. Då definieras

$$\langle f_1, f_2, \ldots, f_s \rangle = \left\{ \sum_{i=1}^{s} h_i \cdot f_i \, : \, h_1, h_2, \ldots, h_s \in k[x_1, x_2, \ldots, x_n] \right\}.$$

Uttrycket $\langle f_1, f_2, \ldots, f_s \rangle$ kallas mängden genererad av $f_1, f_2, \ldots, f_s$, och polynomen f_i kallas **generatorer**.

Sats 1. *([CLO06, sid. 29]) Om $f_1, f_2, \ldots, f_s \in k[x_1, x_2, \ldots, x_n]$, så är $\langle f_1, f_2, \ldots, f_s \rangle$ ett ideal i $k[x_1, x_2, \ldots, x_n]$.*

Bevis. ([CLO06, sid. 29]) Låt $I = \langle f_1, f_2, \ldots, f_s \rangle$. För det första är $0 \in I$, eftersom

$$0 = \sum_{i=1}^{s} 0 \cdot f_i.$$

Låt $f = \sum_{i=1}^{s} h_i \cdot f_i$ och $g = \sum_{i=1}^{s} l_i \cdot f_i$ där $h_i, l_i \in k[x_1, x_2, \ldots, x_n]$. Då gäller:

$$f + g = \sum_{i=1}^{s} (h_i + l_i) \cdot f_i \in I,$$

och för varje $c \in k[x_1, x_2, \ldots, x_n]$,

$$c \cdot f = \sum_{i=1}^{s} (c \cdot h_i) \cdot f_i \in I.$$

Detta visar att I är slutet under addition och multiplikation med polynom, och därmed ett ideal. □

Alla ideal kan genereras av en ändlig uppsättning polynom $f_1, f_2, \ldots, f_s$ så att $I = \langle f_1, f_2, \ldots, f_s \rangle$.

Bevis hänvisas till [CLO06, Sid 74].

Definition 4.19. ([Frö97, s. 48]) Låt $I \subseteq k[x_1, x_2, \ldots, x_n]$ vara ett ideal. Då definieras mängden av dess **ledande monom** som

$$lm(I) = \{lm(f) \, : \, f \in I, f \neq 0\}.$$

Exempel 9. Betrakta polynomen $f_1 = x_1^2 + 2$ och $f_2 = x_1 + x_2$ i $k[x_1, x_2]$. Då är

$$I = \langle f_1, f_2 \rangle = \{f_1 \cdot h + f_2 \cdot g \, : \, h, g \in k[x_1, x_2]\}$$

mängden av alla polynom som kan bildas genom att multiplicera f_1 och f_2 med andra polynom och sedan addera resultaten.

Till exempel gäller:

$$f_1 \cdot 1 + f_2 \cdot 0 = x_1^2 + 2 \in I,$$

$$f_1 \cdot 0 + f_2 \cdot 1 = x_1 + x_2 \in I,$$

$$f_1 \cdot x_2 + f_2 \cdot x_1 = (x_1^2 + 2)x_2 + (x_1 + x_2)x_1 = x_1^2 x_2 + 2x_2 + x_1^2 + x_1 x_2 \in I.$$

Man kan också ta summor och produkter:

$$(x_1^2 + 2) + (x_1 + x_2) = x_1^2 + x_1 + x_2 + 2 \in I,$$

$$(x_1^2 + 2)(x_1 + x_2) \in I.$$

4.5 struktursatsen för binomideal

Sats 2. *Om $f_1 = am_1 + bm_2$ och $f_2 = cm_3 + dm_4$ är binom i $k[x_1, \ldots, x_n]$, där $a, b, c, d \in k$ och m_1, m_2, m_3, m_4 är monom, så är $S(f_1, f_2)$ ett polynom med högst två termer. I synnerhet är $S(f_1, f_2)$ antingen ett binom, en term eller 0.*

Bevis. Ledande termerna är $\mathrm{lt}(f_1) = am_1$ och $\mathrm{lt}(f_2) = cm_3$. Låt $M = \mathrm{mgm}(m_1, m_3)$ där $lm(f_1) = m_1$ och $lm(f_2) = m_3$. Då är

$$S(f_1, f_2) = \frac{M}{am_1}(am_1 + bm_2) - \frac{M}{cm_3}(cm_3 + dm_4).$$

Uttrycket utvecklas till

$$M + b \cdot \frac{M}{m_1} m_2 - M - d \cdot \frac{M}{m_3} m_4.$$

Ledande termerna M och M tar ut varandra, så att

$$S(f_1, f_2) = b \cdot \frac{M}{m_1} m_2 - d \cdot \frac{M}{m_3} m_4.$$

Eftersom både $\frac{M}{m_1} m_2$ och $\frac{M}{m_3} m_4$ är monom, och $b, d \in k$ är koefficienter, består uttrycket av högst två termer. Om dessa termer har samma monomdel, reduceras summan till en term eller till 0. Därför är $S(f_1, f_2)$ antingen ett binom, en term, eller 0. □

Sats 3. *Låt $p = am_1 + bm_2$ vara ett binom, där $a, b \in k$ och m_1, m_2 är monom. Antag att p reduceras med en mängd binom $F = \{f_1, f_2, \ldots, f_l\}$, där varje $f_i = c_i u_i + d_i v_i$ består av två termer. Då är varje steg i reduceringsalgoritmen $R(p, F)$ ett binom, en term eller noll.*

Bevis. Antag att monomordningen är sådan att $m_1 > m_2$, så att $\mathrm{lt}(p) = am_1$.

Om det finns ett $f_i \in F$ vars ledande term är $\mathrm{lt}(f_i) = c_i u_i$ och där $u_i \mid m_1$, reduceras p enligt:

$$q = \frac{am_1}{c_i u_i}, \quad q \cdot f_i = am_1 + d_i q v_i.$$

Då ges

$$p - q \cdot f_i = am_1 + bm_2 - (am_1 + d_i q v_i) = bm_2 - d_i q v_i.$$

Om m_2 och qv_i har samma monomdel, så kombineras termerna till en ny term med koefficient $b - d_i \cdot \frac{a}{c_i}$. Om denna koefficient är 0, reduceras uttrycket till 0; om den inte är 0, återstår en enda term. I övriga fall, då monomdelarna skiljer sig åt, är resultatet ett binom.

Alltså är varje reduktionssteg ett binom, en term eller noll. □

Definition 4.20. Antag flera homogena polynom i n variabler

$$x_1, x_2, \ldots, x_n.$$

Då är $f_1, f_2 \ldots, f_n$ en **fullständig skärning** om ekvationssystemet

$$f_1 = 0$$

$$f_2 = 0$$

$$\vdots$$

$$f_n = 0$$

endast har den triviala lösningen alltså att $x_1 = x_2 = \cdots = x_n = 0$.

Exempel 10. Då $n = 3$ så är variablerna x_1, x_2, x_3 och då är $x_1 + x_2, x_1 + x_3, x_2 + x_3$ en fullständig skärning eftersom att

$$\begin{vmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{vmatrix} \neq 0.$$

Detta visar att polynomen inte är linjärt beroende och därmed endast har den triviala lösningen.

Sats 4. *Det finns*

$$\binom{n-1+d}{n-1}$$

stycken monom av grad d i n variabler.

Bevis. Anta att staplar och stjärnor är utplacerade i någon ordning,

$$*|**.$$

En stjärna som ligger till vänster om stapeln representerar x_1 och en stjärna som ligger höger om stapeln representerar x_2. Exemplet ovan beskriver då monomet $x_1 * x_2^2$. Olika resultat kan åstadkommas med flera eller färre staplar exempelvis

$$***$$

vilket representerar x_1^3 eller

$$*||*|*$$

som representerar $x_1 * x_3 * x_4$. Med hjälp av denna struktur kan den ovannämnda satsen bevisas, då antalet olika monom med graden d och antalet variabler n kan räknas ut utifrån strukturen på staplarna och stjärnorna. Ett monom med n antal variabler kan representeras som $n-1$ antal staplar och om dess grad är d så representeras det med d antal stjärnor. Då finns det

$$\binom{n-1+d}{n-1}$$

sätt att ordna staplarna och stjärnorna på, vilket beskriver antalet monom som existerar.

□

Definition 4.21. Ett **monomideal** är ett ideal I som är genererat av monomen $m_1, m_2, \ldots, m_s$ så att

$$I = \langle m_1, m_2, \ldots m_s \rangle.$$

I är då mängden

$$\{f_1 \cdot m_1 + f_2 \cdot m_2 + \cdots + f_s \cdot m_s \mid f_1, f_2, \ldots, f_s \in k[x_1, x_2, \ldots, x_n]\}.$$

Exempel 11. Antag att

$$I = \langle x_1^2, x_2^2, x_3^2 \rangle$$

då är

$$(x_1 + x_2) \cdot x_1^2 + (0) \cdot x_2^2 + x_1^2 \cdot x_3^2 = x_1^3 + x_1^2 x_2 + x_1^2 x_3^2$$

med i idealet. Notera att de monom som är med i polynomet ovan är delbara med någon av generatorerna.

Sats 5. *Låt I vara monomidealet genererat av monomen*

$$m_1, m_2, \ldots, m_r,$$

då är $f \in I$ om och endast om alla monomen i f delas av något m_i där $i = 1, 2, \ldots, r$.

Bevis. Låt

$$I = \langle m_1, m_2, \ldots, m_r \rangle$$

där varje $m_1, m_2, \ldots, m_r$ är ett monom uppbyggt av variablerna $x_1, x_2, \ldots, x_n$ och ett godtyckligt polynom $f \in k[x_1, x_2, \ldots, x_n]$, så kan relationen bevisas åt vardera håll i två steg. Om $f \in I$, så är

$$f = a_1 \cdot m_1 + a_2 \cdot m_2 + \cdots + a_r \cdot m_r$$

där $a_1, a_2, \ldots, a_r \in k[x_1, x_2, \ldots, x_n]$. Eftersom varje term $a_i \cdot m_i$ bara innehåller monom som är delbara med m_i, följer det att varje monom i f är delbart med något av generatorerna $m_1, m_2, \ldots, m_r$.

För att bevisa åt andra hållet, om varje monom i f är delbara med någon av $m_1, m_2, \ldots, m_r$, så kan f skrivas som

$$f = q_1 \cdot m_1 + q_2 \cdot m_2 + \cdots + q_r \cdot m_r$$

där $q_1, q_2, \ldots, q_r$ är monom eller 0 i $k[x_1, x_2, \ldots, x_n]$. Detta bevisar då att f är med i I då $q_1, q_2, \ldots, q_r \in k[x_1, x_2, \ldots, x_n]$. □

4.6 Hilbertvektorn

Definition 4.22. ([CLO06, sid. 452]) Låt $I = \langle m_1, m_2, \ldots, m_s \rangle$, **Hilbertfunktionen** $H_I(d)$ definieras som antalet monom av grad d i $k[x_1, x_2, \ldots, x_n]$ som inte tillhör I, det vill säga

$$H_I(d) = |M_d \setminus (I \cap M_d)|,$$

där M_d betecknar mängden av alla monom av grad d, och $I \cap M_d$ är mängden av monom i I som har exakt grad d.

Definition 4.23. Hilbertvektorn upp till grad t för ett monomideal I är följden

$$(H_I(0), H_I(1), \ldots, H_I(t)),$$

där varje komponent $H_I(d)$ anger hur många monom av grad d, med $0 \leq d \leq t$ som inte tillhör I.

Sats 6. *Låt*

$$I = \langle x_1^{\alpha_1}, x_2^{\alpha_2}, \ldots, x_n^{\alpha_n} \rangle$$

och sätt

$$t = \sum_{i=1}^{n} \alpha_i - n.$$

Då gäller att varje monom med totalgrad minst $t + 1$ *tillhör* I.

Bevis. Låt

$$I = \langle x_1^{\alpha_1}, x_2^{\alpha_2}, \ldots, x_n^{\alpha_n} \rangle$$

och låt

$$m = x_1^{\beta_1} x_2^{\beta_2} \cdots x_n^{\beta_n}$$

vara ett monom med totalgrad

$$\sum_{i=1}^{n} \beta_i \geq t+1, \quad \text{där} \quad t = \sum_{i=1}^{n} \alpha_i - n.$$

Observera att om $\beta_i < \alpha_i$ för alla i, så är den maximala möjliga totalgraden

$$\sum_{i=1}^{n} (\alpha_i - 1) = \sum_{i=1}^{n} \alpha_i - n = t.$$

Men eftersom vi antar att $\sum \beta_i \geq t+1$, kan det inte gälla att $\beta_i < \alpha_i$ för alla i. Det måste alltså finnas minst ett index i sådant att $\beta_i \geq \alpha_i$, vilket innebär att $x_i^{\alpha_i} \mid m$.

Eftersom I är ett monomideal, följer av Teorem 5 att $m \in I$. □

Exempel 12. Låt $n = 3$ och sätt

$$I = \langle x_1^{d_1}, x_2^{d_2}, x_3^{d_3} \rangle.$$

Då är

$$t = d_1 + d_2 + d_3 - n = d_1 + d_2 + d_3 - 3,$$

och enligt sats 6 tillhör varje monom med totalgrad minst

$$t + 1 = d_1 + d_2 + d_3 - 2$$

ideal I.

4.7 Gröbnerbas

En Gröbnerbas är en uppsättning polynom som strukturerar ett ideal på ett sådant sätt att reduktion med basen ger en entydig normalform för varje polynom. Denna egenskap gör det möjligt att avgöra om ett polynom tillhör idealet, samt att lösa polynomekvationssystem.

Definition 4.24. ([Frö97, sid. 51][CLO06, sid. 74]) Givet en monomordning är en **Gröbnerbas** för ett ideal I en ändlig delmängd $G = \{g_1, g_2, \ldots, g_s\} \subseteq I$ sådan att

$$\langle lm(g_1), lm(g_2), \ldots, lm(g_s) \rangle = \langle lm(I) \rangle,$$

där $lm(f)$ betecknar det ledande monomet i f.

Sats 7. *Låt $f_1, f_2, \ldots, f_n \in k[x_1, \ldots, x_n]$ vara homogena polynom av grad $d_1, d_2, ..., d_n$, och låt $G = \{g_1, g_2, \ldots, g_s\}$ vara en Gröbnerbas för idealet $I = \langle f_1, f_2, \ldots, f_n \rangle$. Då gäller att idealet I är en fullständig skärning enligt Definition 4.20 om och endast om*

$$x_i^{e_i} \in \langle lm(g_1), lm(g_2), \ldots, lm(g_s) \rangle \quad \text{för något } e_i \text{ för varje } i = 1, 2, \ldots, n$$

om och endast om Hilbertvektorn för

$$\langle x_1^{d_1}, x_2^{d_2}, \ldots, x_n^{d_n} \rangle$$

är lika med Hilbertvektorn för

$$\langle lm(I) \rangle.$$

Sats 7 är ett standardresultat i kommutativ algebra, och beviset utelämnas här. Eftersom villkoren är ömsesidigt ekvivalenta kan vilket som helst av dem användas för att verifiera att ett ideal är en fullständig skärning.

Gröbnerbaser är i allmänhet inte entydiga. Beroende på vilka generatorer som används, och i vilken ordning reduktionen sker, kan olika Gröbnerbaser uppstå.

Det visar sig dock att det, för varje ideal och en satt monomordning, finns en särskild Gröbnerbas med vissa minimiegenskaper. Denna kallas den **reducerade Gröbnerbasen**, och vi kommer senare visa att den är entydig. Vi börjar med att definiera den.

Definition 4.25. ([CLO06, Sid 90] [Frö97, Sid 53]) En **reducerad Gröbnerbas** är en Gröbnerbas G för ett ideal I som uppfyller:

1. Varje $g \in G$ är moniskt.
2. För varje $g \in G$, ingen av monomen i g ligger i $\langle lm(G \setminus \{g\}) \rangle$.

Sats 8. *För varje ideal $I \subseteq k[x_1, \ldots, x_n]$ och given monomordning finns exakt en reducerad Gröbnerbas.*

Bevis. Antag att G_1 och G_2 är två olika reducerade Gröbnerbaser för samma ideal I och samma monomordning.

Eftersom båda är Gröbnerbaser för I, gäller att $\langle lt(G_1) \rangle = \langle lt(G_2) \rangle = \langle lt(I) \rangle$. Men eftersom båda dessutom är reducerade, moniska och uppfyller det minimala restvillkoret, så måste varje element i G_1 ha exakt motsvarande element i G_2 och vice versa, annars skulle ett ledande monom i den ena basen kunna dela ett monom i ett annat polynom i den andra basen, vilket strider mot reduceradhetsvillkoret.

Alltså måste $G_1 = G_2$. □

4.8 Buchberger-algoritmen

Buchberger-algoritmen används för att konstruera Gröbnerbaser för ideal i $k[x_1, x_2, \ldots, x_n]$.

Låt $I = \langle f_1, f_2, \ldots, f_m \rangle \neq 0$ vara ett polynomideal. Då kan en Gröbnerbas för I konstrueras i ett ändligt antal steg enligt följande algoritm. Se [Frö97, sid. 59] och [CLO06, sid. 87].

Indata: $F = (f_1, f_2, \ldots, f_m)$
Utdata: En Gröbner bas $G = (g_1, g_2, \ldots, g_s)$ för I, Med $F \subset G$.
$G \leftarrow F$;
upprepa
  $G' \leftarrow G$;
  för *varje par* $\{p, q\}, p \neq q$ *i* G' **gör**
    $S \leftarrow R(S(p, q), G)$;
    om $S \neq 0$ **så**
      $G \leftarrow G \cup \{S\}$;
    slut om
  slutför
tills $G = G'$;

Algorithm 2: Buchberger-algoritmen

Algoritmen terminerar efter ett ändligt antal steg och producerar en Gröbnerbas för idealet. Resultatet är inte entydigt, utan beror på ordningen i indata och valen som görs under reduktionerna. Se [CLO06, sid. 87].

Indata: En Gröbnerbas $G = \{g_1, \ldots, g_t\}$
Utdata: En reducerad Gröbnerbas G'
$G' \leftarrow \emptyset$;
för *varje* $g_i \in G$ **gör**
  $F \leftarrow G \setminus \{g_i\}$;
  $r \leftarrow R(g_i, F)$;
  om $r \neq 0$ **så**
    Gör r moniskt (dela alla termer med ledande koefficienten);
    $G' \leftarrow G' \cup r$;
  slut om
slutför
return G';

Algorithm 3: ReduceradGröbnerbas(G)

Algoritmen hänvisas till [Frö97, s. 54] och [CLO06, s. 90].

Sats 9. *Låt* $G = \{g_1, \ldots, g_t\}$ *vara en Gröbnerbas för idealet* $I = \langle f_1, f_2, \ldots, f_s \rangle$. *Då är* $G' = ReduceradGröbnerbas(G)$ *en reducerad Gröbnerbas.*

Bevis. Eftersom varje g_i' är resultatet av att reducera g_i med avseende på $G - \{g_i\}$, gäller att $g_i - g_i' \in \langle G - \{g_i\} \rangle, \langle G \rangle = I$. Således är $g_i' \in I$, och eftersom detta gäller för alla i, ligger hela G' i I.

Vidare gäller att varje reduktion med $R()$ tar bort alla termer som kan delas av ledande termer från andra polynom. Det innebär att inget monom i något g_i' är delbart med en ledande term i $G' \setminus \{g_i'\}$. Eftersom vi dessutom gör varje polynom moniskt, uppfyller G' villkoren i definitionen av en reducerad Gröbnerbas.

Slutligen visar vi att G' fortfarande är en Gröbnerbas, det vill säga att

$$\langle lm(G') \rangle = \langle lm(I) \rangle.$$

Eftersom varje g_i' är en linjärkombination av polynom i G, och varje term i g_i' är en kombination av termer från G, gäller

$$lm(g_i') \in \langle lm(G) \rangle,$$

således

$$\langle lm(G') \rangle \subseteq \langle lm(G) \rangle.$$

Eftersom G' genererar samma ideal som G, gäller dessutom

$$\langle lm(I) \rangle \subseteq \langle lm(G') \rangle.$$

Sammantaget följer

$$\langle lm(G') \rangle = \langle lm(I) \rangle,$$

vilket visar att G' är en Gröbnerbas. Eftersom G' dessutom är monisk och uppfyller restvillkoret, är det en reducerad Gröbnerbas.

□

4.9 Reduktion med Gröbnerbaser

En Gröbnerbas möjliggör en entydig reduktion av polynom med avseende på ett ideal. Givet ett polynom f och en Gröbnerbas G, kan man utföra reduktion av f enligt algoritm 1 med avseende på G. Resultatet av denna reduktion är ett nytt polynom r, som är enklare än f och inte längre kan reduceras med G.

Definition 4.26. ([Frö97, sid. 53]) En **normalform** av ett polynom f med avseende på ett ideal I och en vald reduktionsmetod är ett polynom r som uppfyller följande villkor:

- $f - r \in I$,
- r är unik med avseende på den använda reduktionsmetoden,
- r är i ett tillstånd där ingen vidare reduktion är möjlig enligt metoden,
- $f \in I$ om och endast om $r = 0$.

Normalformen beror alltså på såväl idealet som på den valda reduktionsalgoritmen.

Sats 10. *([Frö97, sid. 53]) Låt $G = \{g_1, g_2, \ldots, g_s\}$ vara en Gröbnerbas för idealet*

$$I = \langle g_1, g_2, \ldots, g_s \rangle.$$

Då gäller att $R(p_1, G) = R(p_2, G)$ om och endast om $p_1 - p_2 \in I$. Särskilt gäller att $R(p, G) = 0$ om och endast om $p \in I$.

Bevis. ([Frö97, sid. 53]) Antag först att $R(p_1, G) = R(p_2, G) = r$. Då finns polynom $q_1, q_2 \in k[x_1, \ldots, x_n]$ så att

$$p_1 = q_1 + r, \quad p_2 = q_2 + r, \quad \text{med } q_1, q_2 \in I.$$

Det följer att

$$p_1 - p_2 = q_1 - q_2 \in I,$$

eftersom ideal är slutna under subtraktion.

Omvänt, anta att $p_1 - p_2 \in I$. Låt $r_1 = R(p_1, G)$ och $r_2 = R(p_2, G)$. Eftersom $p_i - r_i \in I$ för $i = 1, 2$, så gäller

$$(p_1 - r_1) - (p_2 - r_2) = (p_1 - p_2) - (r_1 - r_2) \in I.$$

Därmed följer att $r_1 - r_2 \in I$.

Men varken r_1 eller r_2 innehåller några termer som är delbara med ledande monom från något $g_i \in G$, så inte heller deras differens kan göra det, utom i fallet när $r_1 - r_2 = 0$. Alltså måste $r_1 = r_2$, och därmed $R(p_1, G) = R(p_2, G)$.

Det sista påståendet följer genom att sätta $p_2 = 0$, vilket ger

$$R(p, G) = R(0, G) = 0 \quad \Leftrightarrow \quad p \in I.$$

□

Sats 11. *Låt $G = \{g_1, g_2, \ldots, g_t\}$ vara en Gröbnerbas för ett ideal $I \subseteq k[x_1, \ldots, x_n]$, och låt $f \in k[x_1, \ldots, x_n]$. Då är resten $r = R(f, G)$ vid division av f med G entydig, oavsett i vilken ordning elementen i G används i divisionen.*

Bevis. Antag att vi utför två olika divisioner av f med G, men i olika ordning, och att detta ger två olika restpolynom r_1 respektive r_2.

Eftersom båda beräkningarna bygger på division med polynom från G, så gäller:

$$f - r_1 \in I \quad \text{och} \quad f - r_2 \in I.$$

Därför följer att

$$(f - r_1) - (f - r_2) = r_2 - r_1 \in I.$$

Samtidigt är både r_1 och r_2 restpolynom i divisionen med G, så inga termer i r_1 eller r_2 är delbara med något ledande monom $\mathrm{lm}(g_i)$ för $g_i \in G$. Detta innebär att även $r_1 - r_2$ inte kan ha några sådana termer.

Men detta innebär att $r_1 - r_2 \in I$ och samtidigt inte kan innehålla några ledande monom från G, vilket enligt definitionen av Gröbnerbas endast är möjligt om $r_1 - r_2 = 0$. Alltså är $r_1 = r_2$, och resten är entydig.

Därmed spelar ordningen i divisionen ingen roll när G är en Gröbnerbas. □

Sats 12. *Låt $G = \{g_1, g_2, \ldots, g_s\}$ vara en Gröbnerbas för idealet $I = \langle G \rangle$. Då är resten $r = R(f, G)$ en **normalform** av f med avseende på idealet I.*

Bevis. För att styrka att $r = R(f, G)$ uppfyller kriterierna i Definition 4.26 verifieras följande punkter:

- *Skillnaden $f - r$ tillhör idealet I*: Detta följer direkt av divisionsalgoritmen. Polynomet f skrivs som en linjärkombination av element i G plus resten r, vilket innebär att $f - r \in I$.
- *Restpolynomet r är entydigt*: Det är visat i sats 11 att om G är en Gröbnerbas, så är resten oberoende av i vilken ordning elementen i G appliceras i divisionen. Därmed är r entydigt bestämt.
- *Ingen vidare reduktion är möjlig*: Divisionen med G avslutas först när inget ledande monom i G delar någon term i r. Därmed är r irreducerbart med avseende på G.
- *Resten är noll om och endast om $f \in I$*: Det har tidigare visats att $R(f, G) = 0$ om och endast om $f \in I$. Detta följer av att alla termer i f då kan reduceras fullständigt av G, vilket endast är möjligt om $f \in I$.

Samtliga krav i Definition 4.26 är därmed uppfyllda. □

Sats 13. *Låt $G = \{g_1, g_2, \ldots, g_s\}$ där $g_1, g_2, \ldots, g_s \in k[x_1, x_2, \ldots, x_n]$ vara en Gröbnerbas för ett ideal I, och låt m vara ett monom av grad d. Då består normalformen av m med avseende på G, under reduktion enligt algoritmen 1 $R(m, G)$, av alla monom av grad d som inte delas av någon ledande term i G eller 0.*

Bevis. Vid varje steg i Buchberger-algoritmen reduceras ledande monom i m med hjälp av ett polynom $g_i \in G$, förutsatt att det ledande monomet i g_i delar någon m. Denna process upprepas tills monomet inte längre är delbart med någon ledande term i G. Alla sådana monom lämnas därför orörda av algoritmen. Det innebär att normalformen enbart består av de monom som inte delas av någon $lt(g)$ för $g_i \in G$ eller 0, vilket var vad som skulle visas. □

4.10 Den alternativa algoritmen

Den alternativa algoritm som används i detta projekt syftar till reduktion av termer i ideal genererade av homogena binom av typen

$$x_i^d - b_i m_i,$$

där m_i är ett monom av totalgrad d och $b_i \in k$. Dessa binom antas generera en fullständig skärning.

I den källa där algoritmen är hämtad [JLN24] behandlas en mer generell klass av binom av formen

$$a_i x_i^d - b_i m_i,$$

där $a_i, b_i \in k$. För att förenkla presentation och implementation används här en specialisering till den moniska formen där $a_i = 1$.

Algoritmen definieras genom en riktad graf $GB_{d,n}$ med hörnmängden

$$\{x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n} : \alpha_1 + \cdots + \alpha_n = d\},$$

och kanter av formen

$$m \xrightarrow{i} \frac{m}{x_i^d} \cdot b_i \quad \text{då } x_i^d \mid m \text{ och } x_j^d \mid m \text{ för alla } j < i.$$

Dessa kanter representerar tillåtna reduktioner där faktorn x_i^d i m ersätts med b_i enligt binomet $x_i^d - b_i$.

Huruvida koefficienter påverkar reduktionsgången undersöks i följande sats.

Sats 14. *Låt $t = c \cdot m$ vara en term där $c \in k$ är en koefficient och m är ett monom. Låt $B = \{(x_i^d, b_i)\}$ vara en ordnad bas av homogena binom som skapar en fullständig skärning. Då påverkar inte koefficienten c vilken reduktionskant som väljs i algoritmsteget*

$$m \xrightarrow{i} \frac{m}{x_i^d} \cdot b_i \quad \textit{då } x_i^d \mid m \textit{ och } x_j^d \mid m \textit{ för alla } j < i.$$

Det följer att algoritmen för monomreduktion kan utvidgas direkt till termreduktion utan att ändra dess struktur.

Bevis. Reduktionssteget i algoritmen baseras enbart på delbarhet av monomet m med x_i^d, samt att inga x_j^d med $j < i$ delar m. Dessa villkor är oberoende av koefficienten c.

Eftersom koefficienten c inte påverkar vilka x_i^d som delar m, kommer samma index i att väljas oavsett c. När reduktionen väl sker utförs följande:

$$t = c \cdot m \longrightarrow c \cdot \left(\frac{m}{x_i^d} \cdot b_i \right),$$

vilket innebär att koefficienten c följer med genom hela beräkningen utan att påverka valet av reduktionssteg.

Därmed påverkar inte koefficienter algoritmens gång, och reduktionsstigen för blir oförändrad. Algoritmen kan därför entydigt utvidgas från monom till godtyckliga termer. □

I denna text används denna struktur som grund för att konstruera en deterministisk algoritm för termreduktion. Vi observerar att koefficienterna i termerna inte påverkar reduktionsgången eftersom de endast multipliceras genom alla steg. Därför utvidgas algoritmen till att verka på godtyckliga termer snarare än enbart monom.

Indata: En term $t = c \cdot m$, där c är en koefficient och m ett monom, samt en uppsättning ordnade homogena binom $B = \{(a_1, b_1 m_1), (a_2, b_2 m_2) \ldots, (a_n, m_2 b_n)\}$ där varje $a_i = x_i^d$
Utdata: En reducerad term r

$r \leftarrow t$;
$S \leftarrow \{\}$;
repeat
  $r' \leftarrow r$;
  om *Monomdelen av* $r \in S$ **så**
    $r \leftarrow 0$;
    bryt loop
  slut om
  $S \leftarrow S \cup \{\text{Monomdelen av } r\}$; **för** $(a_i, b_i m_i) \in B$ **gör**
    om $a_i \mid r$ **så**
      $q \leftarrow r / a_i$;
      $r \leftarrow q \cdot b_i m_i$;
      bryt loop
    slut om
  slutför
until $r = r'$;

Algorithm 4: $AltR(t, B)$

Sats 15. *Algoritmen 4 terminerar efter ändligt många steg och returnerar ett entydigt resultat* $r = AltR(t, B)$ *för varje given term* $t = c \cdot m$.

Bevis. Varje iteration i algoritmen ersätter termen $r = c \cdot m$ med en ny term $r' = c \cdot \frac{m}{x_i^d} \cdot b_i$, där $x_i^d \mid m$, och i är det minsta index så att detta gäller enligt ordningen i basen B. Detta val är deterministiskt, samma a_i kommer alltid att väljas för ett givet m, oberoende av koefficienten c, vilket visas i föregående sats.

Eftersom alla $a_i = x_i^d$ och alla b_i har samma grad d, bevaras totalgraden i varje steg. Det finns därför bara ändligt många monom av totalgrad d, vilket innebär att algoritmen endast kan besöka ändligt många olika termer.

För att undvika oändliga reduktionscykler lagras alla tidigare besökta termer i mängden S. Om algoritmen stöter på en redan besökt term, avbryts processen med $r = 0$. Detta säkerställer att varje körning av algoritmen antingen avslutas i ett irreducerbart tillstånd eller i en cykel som hanteras explicit.

Sammantaget innebär detta att algoritmen alltid terminerar efter ändligt många steg. Eftersom valet av reduktionssteg endast beror på basens ordning och del-

barhet, är algoritmens beteende helt deterministiskt och producerar alltid samma utdata r för varje given term t.

Se även [JLN24] för ett motsvarande resonemang via reduktionsgrafer. □

Sats 16. *Algoritmen 4 terminerar och returnerar $r = AltR(t, B)$, som är en normalform enligt Definition 4.26.*

Bevis. För att visa att $r = \mathrm{AltR}(t, B)$ där $I = \langle B \rangle$uppfyller villkoren i Definition 4.26, beaktas följande:

- $f - r \in I$**:** Varje omskrivning i algoritmen sker enligt $m \mapsto \frac{m}{x_i^d} \cdot b_i$, vilket motsvarar subtraktion av ett element i idealet I. Efter ett ändligt antal sådana omskrivningar gäller att $f - r \in I$.
- **Entydighet:** Algoritmen är deterministisk: för varje term används alltid det första $a_i \mid t$ enligt den fasta ordningen i B. Därmed existerar endast en möjlig reduktionskedja, vilket medför att utdata r är entydig för varje given indata t.
- **Irreducerbarhet:** Reduktionen avslutas när ingen generator a_i i basen B delar termen r. Då kan ingen vidare omskrivning ske, vilket innebär att r är irreducerbar med avseende på algoritmen.
- $f \in I \iff r = 0$**:** Om $f \in I$, leder reduktionsgången till $r = 0$ genom linjärkombinationer av binomer i B. Omvänt, om algoritmen returnerar $r = 0$, innebär det att f helt reducerats med generatorer i I, vilket ger $t \in I$.

Samtliga villkor i Definition 4.26 är därmed uppfyllda. □

Därmed uppfyller $r = \mathrm{AltR}(t, B)$ alla krav i Definition 4.26, och algoritmen ger en normalform.

Sats 17. *Låt $I = \langle x_1^d - m_1,\ x_2^d - m_2,\ \ldots,\ x_n^d - m_n \rangle$, där varje $x_i^d - m_i$ är ett homogent binom av grad d, och m_i är ett monom som inte är delbart med x_i^d. Då är normalformen $r = AltR(t, B)$ antingen 0 eller ett monom där varje variabel x_i förekommer med exponent strikt mindre än d.*

Bevis. Vid varje reduktionssteg ersätts en faktor $x_i^d \mid t$ med ett monom m_i av totalgrad d. Eftersom detta sker endast när exponenten för x_i är minst d, och x_i^d ersätts med ett uttryck där exponenten för x_i är strikt mindre, så minskar någon exponent utan att totalgraden förändras.

Algoritmen avslutas när ingen sådan reduktion är möjlig, vilket innebär att alla variabler förekommer med exponent strikt mindre än d, alltså inget x_i^d delar r.

Om algoritmen upptäcker en cykel returneras $r = 0$, vilket innebär att $t \in I$. Därmed är r antingen noll eller ett monom där inget x_i^d delar r, vilket var att visa. □

Antalet monom av en given grad d som inte tillhör idealet I det vill säga normalformerna är oberoende av vilken reduktionsmetod som används. Både Gröbnerbasreduktion och den alternativa algoritmen ger upphov till samma mängd normalformer, och därmed samma Hilbertfunktion.

Den alternativa algoritmen kan därför användas för att beräkna antalet sådana monom algoritmiskt. Det görs genom att identifiera alla monom av grad d som inte reduceras till 0, det vill säga som inte delas av någon generator x_i^d i basen.

4.11 Exempel på reduktion

Exempel 13. Betrakta termen $t = 3x_1^4$ över kroppen $k = \mathbb{Z}_7$. Reduktionen jämförs mellan två metoder: Gröbnerbasreduktion enligt algoritmen $R(t, G)$, där G är en Gröbnerbas för ett ideal I, samt den alternativa algoritmen $AltR(t, B)$, där $B = \{b_1, \ldots, b_s\}$ är en mängd homogena, moniska binom som genererar $I = \langle B \rangle$.

I detta exempel används fyra variabler ($n = 4$) och grad $d = 2$. Välj binomen på formen

$$x_1^2 - b_1 m_1, \quad x_2^2 - b_2 m_2, \quad x_3^2 - b_3 m_3, \quad x_4^2 - b_4 m_4,$$

där varje m_i är ett monom av grad 2 skapat av variablerna x_1, x_2, x_3, x_4, och $b_i \in \mathbb{Z}_7$. I det aktuella utfallet blir

$$B = \{\, x_1^2 - x_1x_2,\ x_2^2 - 3x_1x_2,\ x_3^2 - 6x_1x_3,\ x_4^2 - 3x_2^2 \,\}.$$

Att $I = \langle B \rangle$ är en fullständig skärning verifieras med sats 7. Beräkna först en reducerad Gröbnerbas G för I med graderad lexikon ordning. Ett möjligt utförande ger

$$\begin{aligned} G = \{&x_3^2x_4^2,\ x_4^4,\ x_1x_4^2,\ x_2x_3^2 + 4x_3x_4^2,\ x_2x_4^2, \\ &x_3^3 + 3x_3x_4^2,\ x_1^2 + 3x_4^2,\ x_1x_2 + 3x_4^2,\ x_1x_3 + x_3^2,\ x_2^2 + 2x_4^2\}. \end{aligned}$$

Det ledande monomidealet blir

$$J_1 = \langle lm(I) \rangle = \langle lm(g_1), \ldots, lm(g_{10}) \rangle = \langle x_3^2x_4^2,\ x_4^4,\ x_1x_4^2,\ x_2x_3^2,\ x_2x_4^2,\ x_3^3,\ x_1^2,\ x_1x_2,\ x_1x_3,\ x_2^2 \rangle.$$

Räkna nu Hilbertvektorn för J_1 upp till grad 4.

- Grad 0: $H_{J_1}(0) = 1$ (endast 1 ligger utanför J_1).
- Grad 1: $H_{J_1}(1) = 4$ (inga generatorer i grad 1).
- Grad 2: av de tio monomen ligger $x_1^2, x_1x_2, x_1x_3, x_2^2$ i J_1; kvar 6. Alltså $H_{J_1}(2) = 6$.
- Grad 3: fyra monom återstår utanför J_1; $H_{J_1}(3) = 4$.
- Grad 4: ett monom återstår; $H_{J_1}(4) = 1$.

Så Hilbertvektorn för (J_1) är lika med $(1, 4, 6, 4, 1)$.

Jämför med monomidealet

$$J_2 = \langle x_1^2,\ x_2^2,\ x_3^2,\ x_4^2 \rangle.$$

Här är

$$HV(J_2) = (1, 4, 6, 4, 1)$$

upp till grad 4. Alltså $HV(\langle lm(I) \rangle) = HV(\langle x_1^2, x_2^2, x_3^2, x_4^2 \rangle)$, vilket enligt sats 7 visar att I är en fullständig skärning.

Steg-för-steg med algoritmen $R(t, G)$:

- **Steg 1:** Kontrollerar om $lt(g_7) = x_1^2$ delar $3x_1^4$: ja, reduceras till $5x_1^2x_4^2$.
- **Steg 2:** Kontrollerar om $lt(g_3) = x_1x_4^2$ delar $5x_1^2x_4^2$: ja, reduceras till 0
- **Slutsats:** $3x_1^4$ reduceras till 0 med hjälp av g_7 och g_3.

Sedan för den alternativa algoritmen **Steg-för-steg med algoritmen $AltR(t, G)$:**

- **Steg 1:** $lt(p_1) = x_1^2$ delar $3x_1^4$, reducerar med binomet (x_1^2, x_1x_2), ger $3x_1^3x_2$.
- **Steg 2:** $lt(p_1) = x_1^2$ delar $3x_1^3x_2$, reducerar med samma binom, ger $3x_1^2x_2^2$.
- **Steg 3:** $lt(p_1) = x_1^2$ delar $3x_1^2x_2^2$, reducerar, ger $3x_1x_2^3$.
- **Steg 4:** $lt(p_2) = x_2^2$ delar $3x_1x_2^3$, reducerar, ger $2x_1^2x_2^2$. Vilket är ett monom som vi redan har sett så vi får att algoritmen slutar i rest $= 0$

Den alternativa och reducering med Gröbnerbas gav båda resultatet 0 vilket visar att x_1^4 är med i idealet I. Gröbnerbasen tog 2 steg medans den alternativa algoritmen tog 4.

Exempel 14. Betrakta monomet $t = x_1^2x_2$ över kroppen $k = \mathbb{Z}_7$. Reduktionen jämförs mellan två metoder: Gröbnerbasreduktion enligt algoritmen $R(t, G)$, där G är en Gröbnerbas för ett ideal I, samt den alternativa algoritmen $AltR(t, B)$, där $B = \{p_1, p_2, p_3\}$ är homogena, moniska binom som genererar $I = \langle B \rangle$.

I detta exempel används tre variabler $(n = 3)$ och grad $d = 2$. Välj

$$p_1 = x_1^2 - 5x_1x_3, \qquad p_2 = x_2^2 - 3x_1^2, \qquad p_3 = x_3^2 - 4x_1^2,$$

så att $B = \{p_1, p_2, p_3\}$ och $I = \langle B \rangle$.

Att $I = \langle B \rangle$ är en fullständig skärning verifieras med sats 7. Beräkna först en reducerad Gröbnerbas G för I.

Beräkna en reducerad Gröbnerbas G för I (graderad lexikografisk ordning, $x_1 > x_2 > x_3$):

$$G = \{\, g_1 = x_3^3,\ g_2 = x_1^2 + 5x_3^2,\ g_3 = x_1x_3 + x_3^2,\ g_4 = x_2^2 + x_3^2 \,\}.$$

Det ledande monomidealet blir

$$J_1 = \langle lm(I) \rangle = \langle x_3^3,\ x_1^2,\ x_1x_3,\ x_2^2 \rangle.$$

Räkna nu Hilbertvektorn för J_1 upp till grad 4.

- Grad 0: $H_{J_1}(0) = 1$ (endast 1 ligger utanför J_1).
- Grad 1: $H_{J_1}(1) = 3$ (inga generatorer i grad 1).
- Grad 2: av de sex monomen ligger x_1^2, x_1x_3, x_2^2 i J_1; kvar 3. Alltså $H_{J_1}(2) = 3$.
- Grad 3: av de tio monomen är alla utom $x_2x_3^2$ multiplar av någon generator; kvar 1. Alltså $H_{J_1}(3) = 1$.
- Grad 4: alla femton monom ligger i J_1; $H_{J_1}(4) = 0$.

Så $HV(J_1) = (1, 3, 3, 1, 0)$.

Jämför med monomidealet

$$J_2 = \langle x_1^2,\ x_2^2,\ x_3^2 \rangle.$$

Direkt kontroll ger

$$HV(J_2) = (1, 3, 3, 1, 0)$$

upp till grad 4. Alltså Hilbertvektorn för $\langle lm(I) \rangle$ är lika med Hilbertvektorn för $HV(\langle x_1^2, x_2^2, x_3^2 \rangle)$, vilket enligt sats 7 visar att I är en fullständig skärning.

Steg-för-steg med algoritmen $R(t, G)$.

- **Steg 1:** $lm(g_1) = x_3^3 \nmid x_1^2x_2$, $lm(g_2) = x_1^2 \mid x_1^2x_2$. Beräkna

$$\frac{x_1^2x_2}{x_1^2} = x_2, \qquad x_1^2x_2 - (x_2 \cdot g_2) = x_1^2x_2 - (x_1^2x_2 + 5x_2x_3^2) = -5x_2x_3^2 \equiv 2x_2x_3^2 \pmod 7.$$

- **Steg 2:** Inget av $lm(g_i) \in \{x_3^3, x_1^2, x_1x_3, x_2^2\}$ delar $2x_2x_3^2$; reduktionen stannar.

Resultat: $R(x_1^2x_2, G) = 2x_2x_3^2$. **Antal steg:** 2. **Steg-för-steg med algoritmen $AltR(t, B)$.**

- **Steg 1:** $x_1^2 \mid x_1^2x_2$. Med $\frac{x_1^2x_2}{x_1^2} = x_2$ och omskrivningen $x_1^2 \mapsto 5x_1x_3$ fås

$$r = x_2 \cdot (5x_1x_3) = 5x_1x_2x_3.$$

- **Steg 2:** Inget av x_1^2, x_2^2, x_3^2 delar $x_1x_2x_3$; reduktionen stannar.

Resultat: $AltR(x_1^2x_2, B) = 5x_1x_2x_3$. **Antal steg:** 2.

Metoderna ger olika normalformer, men båda rester är icke-noll, vilket visar att $x_1^2x_2 \notin I$.

5 Experiment

5.1 Materiel

Experimentet har genomförts med hjälp av egenskriven programkod i Python 3.11.9. All kod har utvecklats specifikt för detta projekt och implementerar algoritmer för polynomreduktion i form av Gröbnerbasberäkning enligt Buchberger samt den alternativa reduktionsalgoritmen.

Inga externa matematikbibliotek som `SymPy` eller `SageMath` har använts för implementationen av samtliga datastrukturer för monom, polynom och ideal då de har implementerats från grunden. Koden av algoritmerna ligger i Appendix A.

För att säkerställa sanningsenlighet i beräkningarna och funktionaliteten i programmets olika delar genomfördes en uppsättning unittester.

Experimenten utfördes i `Jupyter Notebook`, vilket möjliggjorde enkel exekvering av koden, tydlig presentation av resultaten samt smidig hantering och lagring av data.

5.2 Metod

I experimentet sattes den maximala graden till $D = 5$ och antalet variabler till $n = 3$. Kroppen beskrivs som $k = \mathbb{Z}_7$ och polynomringen som $k[x_1, x_2, x_3]$. Monomordningen var graderad lexikografisk ordning.

För varje $\delta \in \{1, 2, \ldots, D\}$ konstruerades ett homogent binomialideal av typen

$$I_\delta = \langle x_1^\delta - c_1 m_1,\ x_2^\delta - c_2 m_2,\ \ldots,\ x_n^\delta - c_n m_n \rangle,$$

där varje $c_i \in k$ och m_i är ett monom av grad δ med $m_i \neq x_i^\delta$. Varje I_δ verifierades vara en fullständig skärning enligt Sats 7 genom att beräkna fram en reducerad Gröbnerbas

$$G_\delta = \{g_{\delta 1}, g_{\delta 2}, \ldots, g_{\delta s}\}$$

med monomordning graderad lexikografisk ordning och kontrollera att $x_i^{e_i} \in \langle lm(g_{\delta 1}), lm(g_{\delta 2}), \ldots, lm(g_{\delta s}) \rangle$ för något e_i för varje $i = 1, 2, 3$. Ifall idealet ej var en fullständig skärning så slumpades ett nytt ideal på samma sätt fram tills att ett ideal med en fullständig skärning hittades. Stegen det tog för att skapa den reducerade Gröbnerbasen sparas.

Därefter konstruerades en gemensam testmängd av monom

$$M_{\leq D} \ = \ \{\, m \in \{x_1, x_2, x_3\}^* \ : \ \text{totalgraden för } m \leq D \,\}.$$

Mängden $M_{\leq D}$ genererades en gång och användes sedan oförändrad mot samtliga ideal I_δ. Eftersom koefficienten inte påverkar reduktionsgången i reduktionerna eller medlemskap i ideal så reducerades enbart monomen.

För varje $\delta \in \{1, 2, \ldots, D\}$ och varje $m \in M_{\leq D}$ utfördes två beräkningar:

1. **Gröbnerbasreduktion:** Resten $R(m, G_\delta)$ beräknades med algoritmen $R(m)$. Antalet reduktionssteg under körningen av $R(m, G_\delta)$ registrerades.
2. **Alternativa algoritmen:** Resten $\mathrm{AltR}(m, B_\delta)$ beräknades, där $B_\delta = \{x_1^\delta - c_1 m_1,\ x_2^\delta - c_2 m_2,\ \ldots,\ x_n^\delta - c_n m_n\}$. Antalet reduktionssteg registrerades.

Resultaten sammanställdes med avseende på (i) medelantal reduktionssteg över hela testmängden $M_{\leq D}$ och samtliga ideal I_δ, $\delta = 1, \ldots, D$; (ii) medelantal reduktionssteg per idealgrad δ; samt (iii) medelantal reduktionssteg per monom $m \in M_{\leq D}$ (ordnade enligt graderad lexikografisk ordning). För Gröbnerbasreduktion redovisas både körningskostnaden för $R(m, G_\delta)$ och den totala kostnaden där även stegen för konstruktion av G_δ (Buchberger följt av reducering till reducerad Gröbnerbas) inkluderas. Normalformer bestäms av respektive rest $R(m, G_\delta)$ respektive $\mathrm{AltR}(m, B_\delta)$.

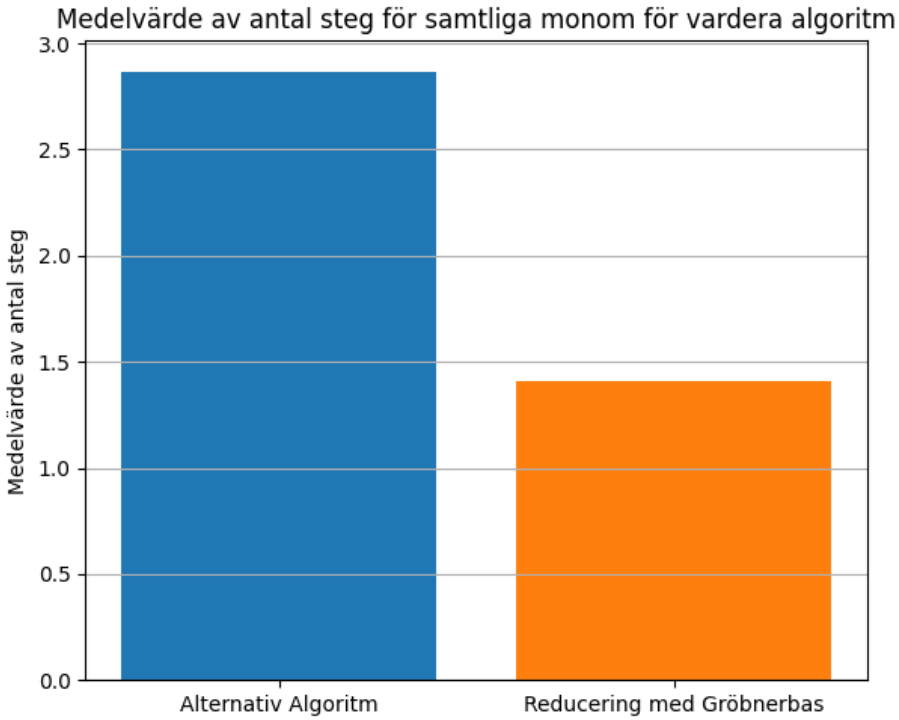
Detta gjordes tre gånger då med samma D, n, k och polynomordning. De tre olika gångerna slumpades olika set av ideal fram.

6 Resultat

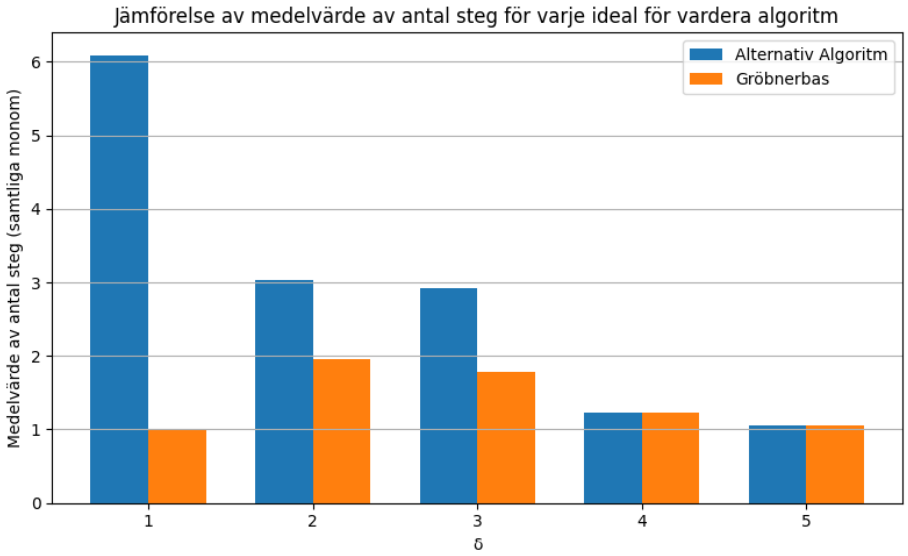
Givet de parametrar som angavs ovan, det vill säga antalet variabler $n = 3$ i polynomringen, kroppen $k = \mathbb{Z}_7$ samt den maximala graden $D = 5$ och graderad lexikografisk ordning för skapandet av Gröbnerbaserna, redovisas de ideal som var en fullständig skärning som slumpades fram av experimentet i varje tabell. Detta repeterades tre gånger och visas efter varandra.

grad δ	Ideal
1	$\langle x_1 - x_2, x_1 - 2x_3, x_3 - 2x_1 \rangle$
2	$\langle x_1^2 - 3x_1x_2, x_2^2 - 4x_1x_2, x_3^2 - 6x_1x_3$
3	$\langle x_1^3 - 5x_2^2x_3, x_2^3 - 6x_3^3, x_3^3 - 2x_1^2x_2 \rangle$
4	$\langle x_1^4 - 4x_1^2x_3^2, x_2^4 - 3x_1x_2^2x_3, x_3^4 - 3x_1^3x_2 \rangle$
5	$\langle x_1^5 - 4x_1^3x_2^2, x_2^5 - 5x_1^4x_2, x_3^5 - 2x_2^2x_3^3 \rangle$

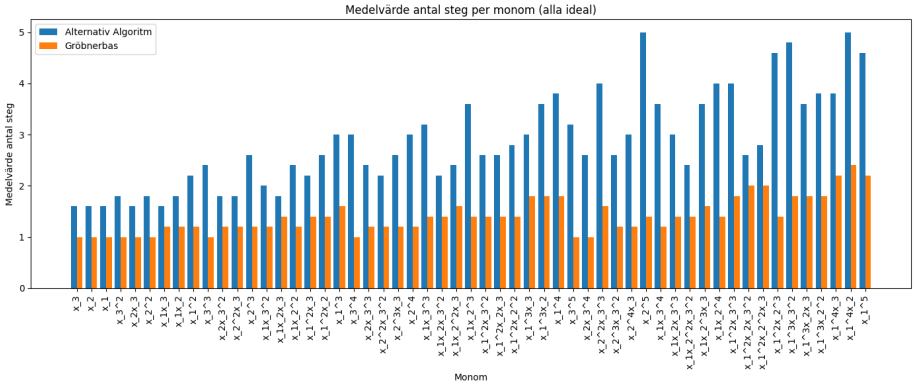
Tabell 1: Ideal (Försök 1)



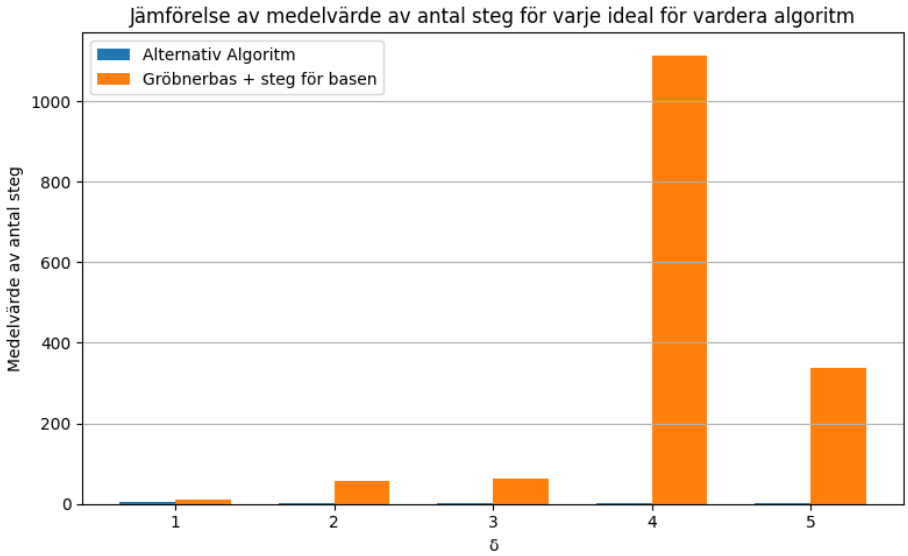
Figur 1: Medelvärdet av antal steg för alla reduceringar (Försök 1)



Figur 2: Medelvärdet av antal steg för varje ideal (Försök 1)



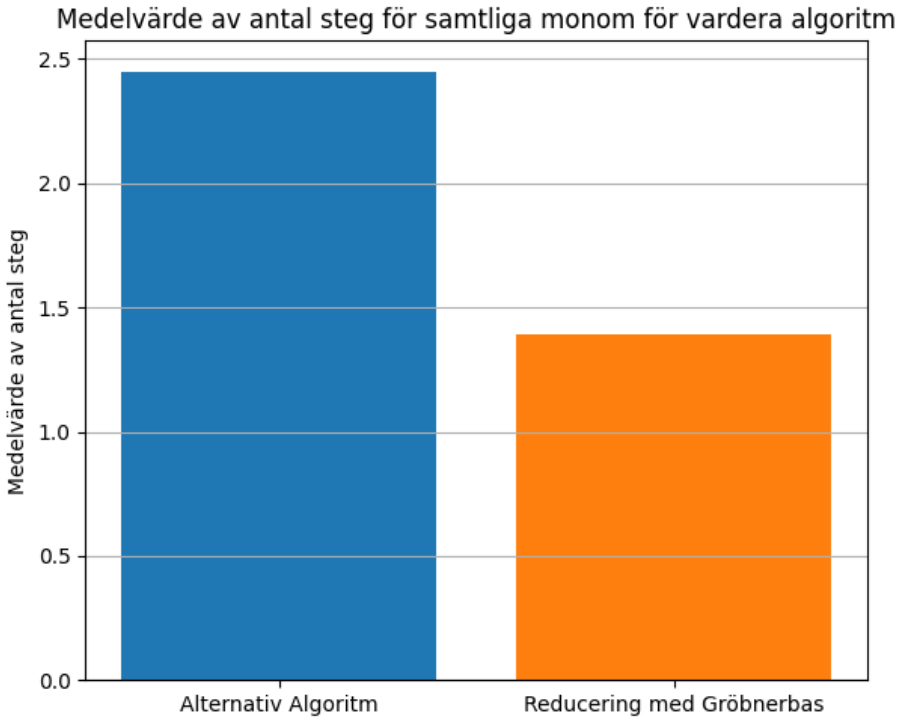
Figur 3: Medelvärdet av antal steg för varje monom (Försök 1)



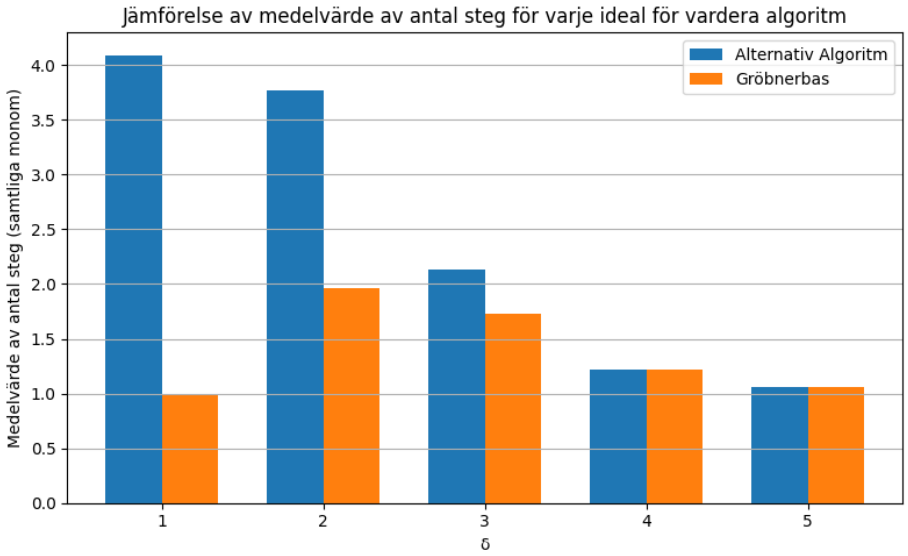
Figur 4: Medelvärdet av antal steg för vardera ideal inkluderade skapandet av Gröbnerbaserna (Försök 1)

grad δ	Ideal
1	$\langle x_1 - 3x_3, x_2 - x_3, x_3 - 3x_2 \rangle$
2	$\langle x_1^2 - 6x_1x_2, x_2^2 - 6x_1^2, x_3^2 - 2x_1x_3$
3	$\langle x_1^3 - 3x_2^3, x_2^3 - 6x_2^2x_3, x_3^3 - 2x_2^2x_3 \rangle$
4	$\langle x_1^4 - 5x_1^2x_3^2, x_2^4 - x_1^2x_3^2, x_3^4 - x_1x_2x_3^2 \rangle$
5	$\langle x_1^5 - 3x_2x_3^4, x_2^5 - 6x_1^2x_2^3, x_3^5 - 4x_2^4x_3 \rangle$

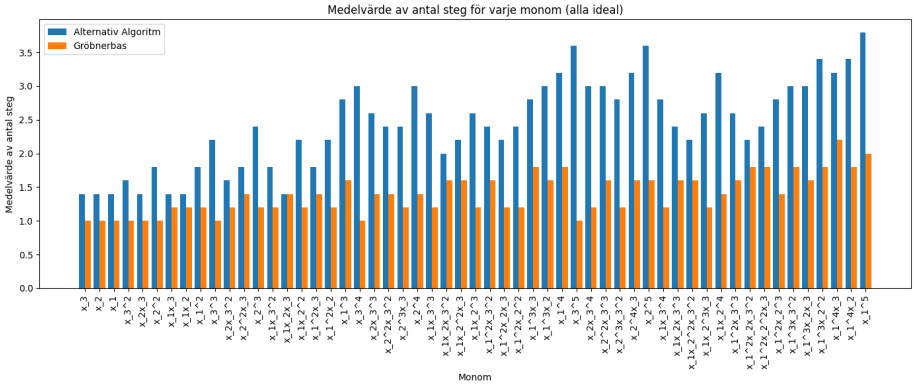
Tabell 2: Ideal (Försök 2)



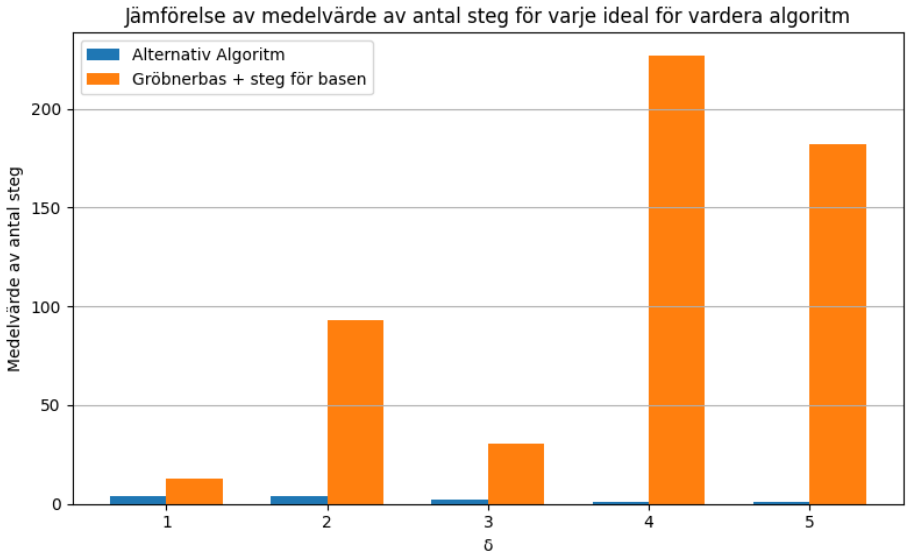
Figur 5: Medelvärdet av antal steg för alla reduceringar (Försök 2)



Figur 6: Medelvärdet av antal steg för varje ideal (Försök 2)



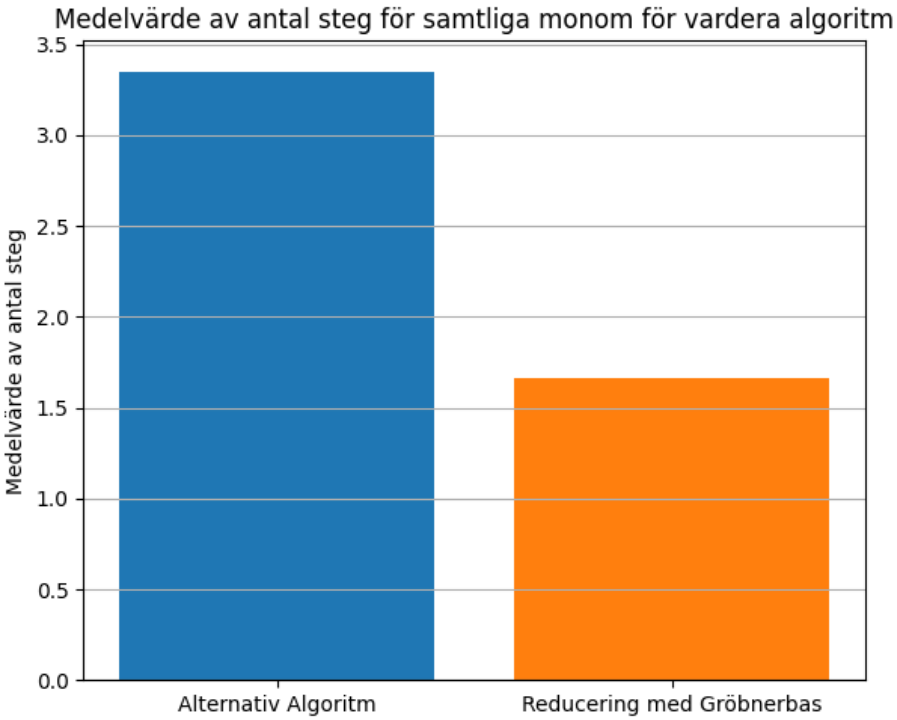
Figur 7: Medelvärdet av antal steg för varje monom (Försök 2)



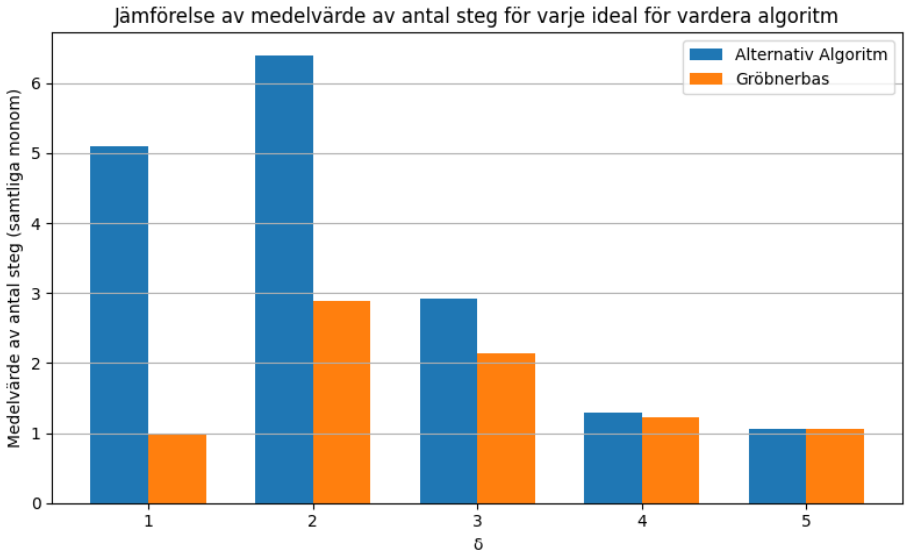
Figur 8: Medelvärdet av antal steg för vardera ideal inkluderade skapandet av Gröbnerbaserna (Försök 2)

grad δ	Ideal
1	$\langle x_1 - 5x_3, x_2 - 4x_1, x_3 - 5x_1 \rangle$
2	$\langle x_1^2 - x_2^2, x_2^2 - x_2x_3, x_3^2 - 3x_1x_2$
3	$\langle x_1^3 - 2x_2^2x_3, x_2^3 - 2x_1x_3^2, x_3^3 - 2x_1^3 \rangle$
4	$\langle x_1^4 - 2x_1x_2^2x_3, x_2^4 - 2x_1^2x_3^2, x_3^4 - x_2^4 \rangle$
5	$\langle x_1^5 - 6x_1x_2^4, x_2^5 - 5x_1^4x_3, x_3^5 - 2x_1^2x_3^3 \rangle$

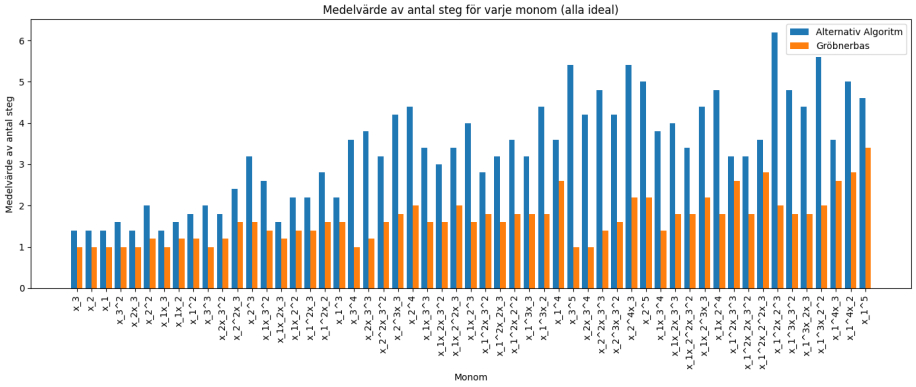
Tabell 3: Ideal (Försök 3)



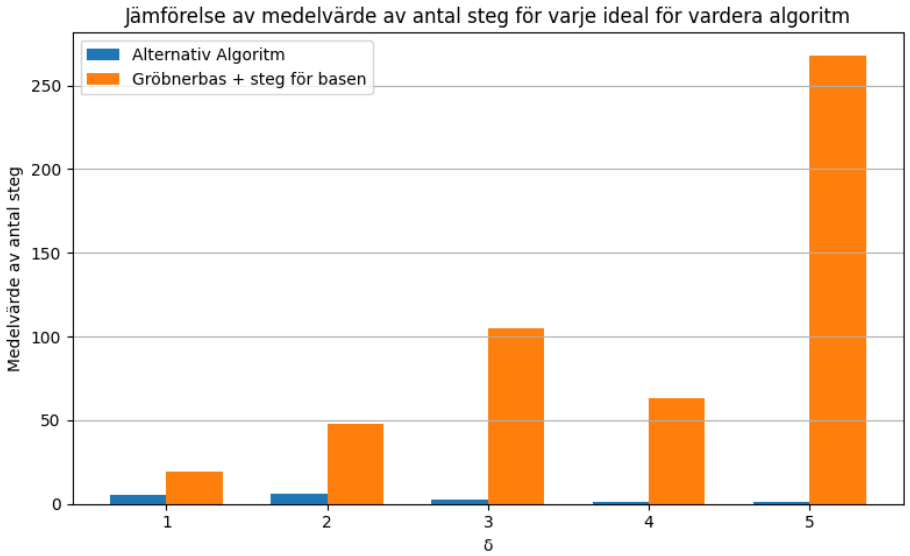
Figur 9: Medelvärdet av antal steg för alla reduceringar (Försök 3)



Figur 10: Medelvärdet av antal steg för varje ideal (Försök 3)



Figur 11: Medelvärdet av antal steg för varje monom (Försök 3)



Figur 12: Medelvärdet av antal steg för vardera ideal inkluderade skapandet av Gröbnerbaserna (Försök 3)

Tolkning av figurer

Figurerna 1, 5, 9(Medelvärdet av antal steg för alla reduceringar). Figurerna visar de två metoderna över hela testmängden. Varje stapel visar medelantal reduktionssteg beräknat över samtliga ideal I_δ, $\delta = 1, 2, \ldots, D$, samt över alla testmonom $m \in M_{\leq D}$. För Gröbnerbasreduktion används resten $R(m, G_\delta)$ och för den alternativa metoden används $\mathrm{AltR}(m, B_\delta)$. Antalet steg för att konstruera G_δ ingår inte i denna figur.

Figurna 2, 6, 10(Medelvärdet av antal steg för varje ideal). Här redovisas medelantalet reduktionssteg för varje ideal med $\delta = 1, 2, 3, 4, 5$. För varje δ beräknas medelvärdet över alla $m \in M_{\leq D}$ för respektive metod. Inte heller här ingår stegen att beräkna G_δ, utan figuren hanterar enbart körningen av $R(m, G_\delta)$ respektive $\mathrm{AltR}(m, B_\delta)$.

Figurerna 3, 7, 11 (Medelvärdet av antal steg för varje monom). Dessa figurer visar, för varje enskilt testmonom $m \in M_{\leq D}$, medelantal reduktionssteg beräknat över alla ideal $\delta = 1, 2, \ldots, D$. Varje par staplar visar de två metodernas medelvärden för samma m.

Figur 4, 8, 12 (Medelvärdet antal steg för varje ideal där Gröbner har med stegen det tog att skapa Gröbnerbasen). Figurerna motsvarar

figurerna 2, 6, 10 men med stegen för att konstruera den reducerade Gröbnerbasen inkluderad för Gröbnermetoden det vill säga antalet steg som krävs för att konstruera G_δ med Buchberger-algoritmen och därefter att göra den till en reducerad Gröbnerbas, adderat med körningskostnaden för $R(m, G_\delta)$. För den alternativa algoritmen $\mathrm{AltR}(m, B_\delta)$ är $B_\delta = \{x_1^\delta - c_1 m_1,\ x_2^\delta - c_2 m_2,\ \ldots,\ x_n^\delta - c_n m_n\}$ redan givet, så någon motsvarande basbyggnadskostnad tillkommer inte. Staplarna visar alltså total stegen för varje δ för respektive metod.

Samtliga medelvärden beräknas över den gemensamma testmängden $M_{\leq D}$ och över de slumpade fullständiga skärningarna I_δ för $\delta = 1, 2, \ldots, D$. Normalformer avgörs enligt respektive metods rest ($R(m, G_\delta)$ eller $\mathrm{AltR}(m, B_\delta)$).

7 Diskussion

I Figur 1 så visas det att den alternativa algoritmen generellt kräver fler reduktionssteg än algoritmen baserad på Gröbnerbaser. Det är viktigt att notera att mätning här avser antalet steg efter att Gröbnerbasen redan har beräknats. Själva konstruktionen av Gröbnerbasen som ofta är beräkningsmässigt tung har inte inkluderats i mätningen.

I 4 kan vi se att den alternativa algoritmen använder färre steg och därmed mindre arbete då man räknar med arbetet det tog att skapa Gröbnerbasen till idealen.

Vidare bör det påpekas att den version av Buchberger-algoritmen som använts i detta arbete inte är optimerad. Det finns förbättrade varianter som kan minska antalet nödvändiga beräkningar avsevärt.

Det är även inte den mest effektiva reduceringen med gröbnerbaser även om vi använder här just en reducerad gröbnerbas så finns det flera knep som man kan använda sig av för att göra att reduceringen går snabbare och hoppa över onödiga steg.

I denna studie har vi medvetet valt att inte mäta faktisk exekveringstid, eftersom sådana jämförelser kräver implementation i identiska programmeringsmiljöer och optimering av båda algoritmerna på låg nivå. Istället har vi valt att använda antalet reduktionssteg för beräkningsarbete.

8 Slutsats

Resultaten visar att den alternativa algoritmen generellt kräver fler reduktionssteg än Gröbnerbas-algoritmen när endast själva reduktionen av monom beaktas. Det är dock viktigt att notera att Gröbnerbas-metoden förutsätter att en Gröbnerbas redan har beräknats, vilket i sig är en beräkningsmässigt kostsam process. Om man även inkluderar stegen för att konstruera Gröbnerbasen i den totala arbetsinsatsen, förändras jämförelsen markant. I figur 4 framgår att den

alternativa algoritmen i många fall leder till färre totala steg och därmed mindre arbete när hela processen beaktas.

Det är också värt att påpeka att den Gröbnerbas-algoritm som använts i denna studie är en grundläggande version av Buchberger-algoritmen, som inte är den mest effektiva möjliga. Det finns flera förbättringar och optimeringar som kan implementeras för att minska antalet steg och beräkningstid för Gröbnerbaser. Vidare har denna studie fokuserat på antalet reduktionssteg snarare än faktisk körtid, eftersom en rättvis jämförelse av körtider skulle kräva implementationer i samma programmeringsspråk och under likvärdiga förhållanden.

Sammanfattningsvis visar resultaten att Gröbnerbas-reduktion är effektivare för enskilda reduktioner, men att den alternativa algoritmen kan vara fördelaktig när man tar hänsyn till hela arbetsflödet, särskilt i situationer där konstruktionen av Gröbnerbasen är mycket kostsam relativt antalet reduktioner som ska utföras.

Referenser

[CLO06] David Cox, John Little och Donal O'Shea. *Ideals, Varieties, and Algorithms*. Springer Cham, 2006.

[Frö97] Ralf Fröberg. *An Introduction to Gröbner Bases*. Springer Chamg, 1997.

[JLN24] Filip Jonsson Kling, Samuel Lundqvist och Lisa Nicklasson. "On binomial complete intersections". I: *Journal of Algebra* 649 (2024), s. 12–34. ISSN: 0021-8693. DOI: https://doi.org/10.1016/j.jalgebra.2024.03.012. URL: https://www.sciencedirect.com/science/article/pii/S0021869324001467.

A Python kod

```python
def alternativ_algorithm(monomial: Poly,
    basis: list[tuple[Poly,Poly]], test=False) -> Poly:
    """
    Reduce a monomial with respect to a basis of polynomials using an alternative algorithm.

    Args:
        monomial (Poly): The monomial to reduce.
        basis (list[tuple[Poly,Poly]]): The basis of binomials to reduce against.
            Saved as a list of tuples where each tuple contains two monomials (binomials).
    """

    m = monomial.copy()
    steps = 0
    divisble_term = None
    seen_monomials = [m]
    basis.sort(key=lambda x: x[0], reverse=True)
    r = None
    while m != 0 and m != r:
        steps += 1
        r = m.copy()
        divisble_term = None
        k = 0
        logging.debug(f"Current monomial: {str(m)}")
        for binom in basis:
            k += 1
            ltb = binom[0]
            term = m.copy()
            try:
                if term != 0:
                    remainder = None
                    try:
                        remainder = term / ltb
                        logging.debug(
                            f"Checking divisibility: {str(term)} / {str(ltb)} = {str(remainder)}"
                        )
                    except ValueError:
                        remainder = None
                    if remainder is not None:
                        divisble_term = term
                        break
            except ValueError:
                continue

            if divisble_term:
                break
        if not divisble_term:
            logging.debug(
```

```
                "Ingen ytterligare reduktion är möjlig. Polynomet är redan i sin reducerade form."
            )
            break
        else:
            logging.debug(f"binomials to choose from: {str(basis)}")
            logging.debug(f"Reducing with binomial p_{k}={str(binom)}")
            ltb = binom[0]
            logging.debug(f"Leading term of binomial: {str(ltb)}")
            logging.debug(f"Divisible term: {str(divisble_term)}")
            logging.debug(f"r/ltb = {str(m)} / {str(ltb)}")
            fraction = m / ltb
            logging.debug(f"Fraction: {str(fraction)}")

            m = fraction * binom[1]
            logging.debug(f"r = fraction * b_2 = {str(m)}")
            logging.debug("----------------")
            divisble_term = None
        if m._leading_monomial_id in seen_monomials:
            logging.debug("Monomial has been seen before, stopping reduction.")
            m = 0
        else:
            seen_monomials.append(m._leading_monomial_id)
    logging.debug(f"Final reduced monomial: {str(m)}")
    logging.debug(f"Reduction steps: {steps}")
    if test:
        logging.debug(f"Test mode active.")
        return {"remainder": m, "steps": steps}
    return m
```

```
    def compute_groebner_basis(F: list[Poly], test = False) -> np.ndarray:
    """
    Compute the Groebner basis for a set of polynomials.
    """
    logging.debug("Computing Groebner basis...")
    G = F.copy()
    Gprim = []
    update = 0
    checked_zero_pairs = set()  # Spara endast par som reduceras till noll
    if test:
        logging.info("Testing mode activated.")
        steps = 0
    while G != Gprim:
        Gprim = G.copy()
        L = []
        n = len(G)
        for i in range(n - 1):
            for j in range(i + 1, n):
                if (i, j) not in checked_zero_pairs:
                    L.append((i, j))
```

```
            l_update = len(L)
            for (a, b) in L:
                poly_p = G[a]
                poly_q = G[b]
                # Reduce the S-polynomial with respect to the current Groebner basis
                logging.debug(f"Current Groebner basis: {[str(g) for g in G]}")
                logging.info(f"We have {len(G)} polynomials in the Groebner basis.")
                logging.info(f"We have checked {update} pairs of polynomials.")
                logging.info(f"There is currently {l_update} pairs of polynomials to check.")
                l_update -= 1
                update += 1
                logging.debug(f"Computing S-polynomial for polynomials {a + 1} and {b + 1}")
                S = s_polynomial(poly_p, poly_q)
                result = reduce_polynomial(S, G, test=test)
                if test:
                    steps += result["steps"]
                    S = result["remainder"]
                else:
                    S = result
                if S != 0:
                    logging.debug(f"Reduced polynomial: {S}")
                    G.append(S)
                    logging.debug(f"Updated G: {[str(g) for g in G]}")
                    logging.debug(f"Updated pairs L: {[(i + 1, j + 1) for i, j in L]}")
                checked_zero_pairs.add((a, b))
    G = reduce_basis(G)
    print(f"Gröbnerbasen är: {G}")
    if test:
        logging.info(f"Total steps taken: {steps}")
        return {"basis": G, "steps": steps}
    return G

def s_polynomial(poly_p: Poly, poly_q: Poly):
    """
    Compute the S-polynomial of two polynomials.
    Args:
        poly_q (Poly): The first polynomial.
        poly_p (Poly): The second polynomial.
    Returns:
        Poly: The S-polynomial of the two polynomials.
    """
    lm_p = Poly.lm(poly_p) # i
    lm_q = Poly.lm(poly_q) # j
    lt_p = Poly.lt(poly_p) # i
    lt_q = Poly.lt(poly_q) # j
    Lcm = Poly.lcm(lm_q, lm_p)

    S = ( (Lcm / lt_p) * poly_p ) - ( (Lcm / lt_q) * poly_q )
    logging.debug(f"Leading term of polynomial {str(poly_p)}: {lm_p}")
    logging.debug(f"Leading term of polynomial {str(poly_q)}: {lm_q}")
```

```
71      logging.debug(f"Least common multiple of leading terms: {Lcm}")
72      logging.debug(f"S-polynomial before reduction: {S}")
73      return S
74
75  def reduce_basis(basis: list[Poly]) -> list[Poly]:
76      """
77      Takes a list of Poly objects 'basis' (a Groebner basis) and returns a reduced Groebner basis.
78
79      - Each f_i is reduced with respect to the polynomials already chosen in 'output'
80        (not against the whole original basis).
81      - If the remainder is zero: discard f_i.
82      - Otherwise: normalize so that the leading coefficient = 1, and add it to 'output'.
83
84      Args:
85          basis (list[Poly]): The initial Groebner basis to be reduced.
86
87      Returns:
88          list[Poly]: The reduced Groebner basis.
89      """
90      output: list[Poly] = []
91      i = 0
92      print(f"Original basis:")
93      print(f"{[str(f) for f in basis]}")
94      basis.sort()
95      print("sorted basis:")
96      print(f"{[str(f) for f in basis]}")
97      for f in basis:
98          others = basis[:i] + basis[i+1:]
99          i += 1
100         f_prim = reduce_polynomial(f, output)
101         if f_prim == 0:
102             continue
103         lt_idx = f_prim._leading_monomial_id
104         lc_coeff = f_prim._expr_as_list[lt_idx]
105         # 4) Inversen av lc_coeff i Z/p
106         inv_lc = pow(lc_coeff, -1, Poly.domain)
107         new_coeffs = [(c * inv_lc) % Poly.domain for c in f_prim._expr_as_list]
108         f_prim_monic = Poly(new_coeffs)
109         output.append(f_prim_monic)
110     output.sort(reverse=True)
111     return output
112
113 def reduce_polynomial(
114     polynomial: Poly,
115     basis: list[Poly],test: bool = False,
116 ) -> Poly:
117     """
118     Reduce a polynomial with respect to a Groebner basis.
119     Args:
120         polynomial (Poly): The polynomial to reduce.
```

```
        basis (list[Poly]): The Groebner basis.
        variables (list[str]): List of variable names.
        domain (str): The domain of the coefficients.
        order (str): The monomial order ('lex', 'grlex', 'grevlex').
    Returns:
        Poly: The reduced polynomial.
    """
    if test:
        steps = 0
    if polynomial != 0:
        p = Poly(polynomial._expr_as_list)
    else:
        p = 0
    logging.debug(f"Reducing polynomial: {p}")
    r = 0
    while p != 0:
        if test:
            steps += 1
        k = 0
        found_divisible = False
        for g in basis:
            k += 1
            ltg = Poly.lt(g)
            ltp = Poly.lt(p)
            logging.debug(f"Checking polynomial {k}: {g} with leading term {ltg}")
            try:
                q_term = ltp/ltg
                logging.debug(f"{p} is divisible by leading term {ltg}: {q_term}")
                p -= q_term * g
                found_divisible = True
                break
            except ValueError as e:
                logging.debug(f"{p} is not divisible by leading term {ltg}: {e}")
        if not found_divisible:
            ltp = Poly.lt(p)
            r = ltp + r  # addera nästkommande ledande termer
            p = p - ltp
            logging.debug(f"      Subtracted LT(p); new p = {p}")
    logging.debug(f"Reduction complete; remainder = {r}")
    if test:
        return {"remainder": r, "steps": steps}
    return r


if __name__ == "__main__":
    logging.info("Startar groebner_basis_matrix.py")
    variables = ['x', 'y']
```

- **I exempel 10** ska det stå att exemplet fungerar för linjära polynom inte polynom som har högre grad än 1. Alltså inte till exempel:

  $$p_1 = x_1, p_2 = x_2, p_3 = x_1x_3$$

  är linjärt oberoende men de är de har inte bara den triviala lösningen utan

  - $x_1 = 0$
  - $x_2 = 0$
  - $x_1x_3 = 0$

  visar att det finns oändligt många lösningar som är $\{(0, 0, t) : t \in k\}$.

- **Stycket innan sats 14** ska det stå att i artikeln [JLN24] så behandlades

  $$a_i x_i^{d_i} - b_i m_i$$

  och det ska stå

  $$m \xrightarrow{i} \frac{m_i m}{x_i^{d_i}} \quad \text{då } x_i^{d_i} \mid m \text{ och } x_j^{d_j} \nmid m \text{ för alla } j < i.$$

  Så det som var skillnaden på algoritmen som jag använde mig av var att alla binom hade samma grad d och att $a_i = 1$.

- **I sats 14** ska det stå

  $$B = \{(x_i^{d_i}, b_i m_i)\}$$

  och

  $$m \xrightarrow{i} \frac{m_i m}{x_i^{d_i}} \cdot b_i \quad \text{då } x_i^{d_i} \mid m \text{ och } x_j^{d_j} \nmid m \text{ för alla } j < i.$$

- **I Algoritm 4** så står det $a_i = x_i^d$ men det är tydligare med att ha kvar x_i^d då innan användesa_i som en koefficient.