並列処理を活用した複数の包括的グレブナ基底系アルゴリズムの融合について

Merging two algorithms for computing comprehensive Gröbner systems by utilizing parallel processing

東京理科大学大学院理学研究科　　和田夏[*1]
Natsu Wada
Graduate School of Science, Tokyo University of Science

東京理科大理学部第一部応用数学科　　鍋島克輔[*2]
Katsusuke Nabeshima
Department of Applied Mathematics, Tokyo University of Science

Abstract

Two algorithms for computing comprehensive Gröbner systems are merged by utilizing parallel processing. A new framework for computing comprehensive Gröbner systems is given by using the multi-core architecture of modern computers and combining parallel processing through the computer algebra system Risa/Asir and the OpenXM Asir server. The effectiveness of the approach is also discussed.

1　はじめに

本稿では，複数の包括的グレブナ基底系計算アルゴリズムを並列計算機能を用いて融合し，その効率性について計算実験を行う．

包括的グレブナ基底系はパラメータ付きイデアルの解析に役立つ．包括的グレブナ基底系計算アルゴリズムは大きく分けて 4 つ存在する．初めて包括的グレブナ基底が紹介された 1992 年の Weispfenning の論文 [18] では S 多項式を計算する際，先頭項がゼロがゼロでないかで分けて計算する方法が提案された．2001 年には Suzuki-Sato の Alternative comprehensive Gröbner bases (ACGB)[15] において，von Neumann 正規環を係数ドメインとする多項式環のイデアルのグレブナ基底と包括的グレブナ基底系を同一視できることが示された．さらに，2006 年，Suzuki-Sato[17] によりブロック項順序を用いて多項式環を係数ドメインとする多項式環のイデアルのグレブナ基底を繰り返し行う方法が紹介され，これはグレブナ基底の安定性理論を用いた画期的なものであり，アルゴリズムのシンプルさと計算効率の良さが目立ったものである．そして，2024 年 Nabeshima[11] ではイデアルの安定性理論を用い，有理関数体を係数とするグレブナ基底計算とイデアル商計算を繰り返し行う方法が発表された．これらの計算アルゴリズムはそれぞれ改良され続け，より良いものへと研究が進んでいるが，包括的グレブナ基底系の計算時間は問題とアルゴリズムに依存する．

[*1] 〒 162-0825 東京都新宿区神楽坂 1-3　E-mail: 1422547@ed.tus.ac.jp
[*2] 〒 162-0825 東京都新宿区神楽坂 1-3　E-mail: nabeshima@rs.tus.ac.jp

そこで，この 4 つの包括的グレブナ基底系アルゴリズムの中でも，イデアルの安定性理論を用いた Suzuki-Sato と Nabeshima のアルゴリズムが速いと言われているため，それらの融合を試みた．現在のコンピュータは複数のコアを持っているため，それらを用いて複数のアルゴリズムを並列計算を用いて融合する．本研究では，複数アルゴリズムを融合したプログラムを複数作成し，その有効性について解析を行った．また，ゼロ次元での計算に特化している 2003 年 Sato-Suzuki-Nabeshima[14] と 1997 年の Kalkbrener の結果も用いた．前者は 2007 年に Kurata-Noro[5] によりすでにプログラムが存在するのでそれを使用した．

2 準備

本稿では, 次の記号を使う．$\mathbb{Q}$ を有理数体とし, $\mathbb{C}$ を複素数体とする．$x = x_1, \ldots, x_n$ を変数，$t = t_1, \ldots, t_m$ をパラメータとし，また $x \cap t = \emptyset$ とする．K を体 ($\mathbb{Q}$ を想定) とし，L は K を含む代数的閉体 ($\mathbb{C}$ を想定) とする．また， $\mathrm{pp}(x), \mathrm{pp}(t), \mathrm{pp}(t, x)$ をそれぞれの変数集合 $x, t, t \cup x$ からなる項の集合とし, 項順序をペア $(\prec_t, \prec_x)$ で表したものは $t \ll x$ となるブロック項順序を意味し, $\mathrm{pp}(t)$ にお ける項順序は $\prec_t$ であり，変数 $\mathrm{pp}(x)$ における項順序は $\prec_x$ となる．$f \in K[t][x]$ (係数が多項式環 $K[t]$ にあり主変数は x) のとき項順序 $\prec_x$ に関して f の先頭項, 先頭係数, 先頭単項をそれぞれ $\mathrm{lt}_x(f), \mathrm{lc}_x(f), \mathrm{lm}_x(f)$ で表す．また, 多項式 f は $K[t, x]$ の元とみなすこともできるので, この場合, 項順序 $\prec_{t,x}$ における f の先頭項, 先頭係数, 先頭単項をそれぞれ $\mathrm{lt}_{t,x}(f), \mathrm{lc}_{t,x}(f), \mathrm{lm}_{t,x}(f)$ で表す．F を $K[t][x]$ の部分集合とする．このとき, $\mathrm{lc}_x(F) := \{\mathrm{lc}_x(f) \mid f \in F\} \subset K[t], \mathrm{lt}_x(F) := \{\mathrm{lt}_x(f) \mid f \in F\}$ とする．$f_1, \ldots, f_l \in R$ としたとき (R は単位元を持つ可換環), 記号 $\langle \cdot \rangle$ の定義を $\langle f_1, \ldots, f_l \rangle := \left\{ \sum_{i=1}^{l} h_i f_i \mid h_1, \ldots, h_l \in R \right\}$ とする．

任意の元 $\bar{a} \in L^m$ に対して, 特化準同型写像 $\sigma_{\bar{a}} : K[t] \to L$ を定義する. もちろん，この写像は自然な拡張として $\sigma_{\bar{a}} : K[t][x] \to L[x]$ とも考えることができるのでこの意味としても同じ記号を用いる. 写像 σ でのイデアル $I \subset K[t][x]$ における像は $\sigma(I) := \{\sigma(f) \mid f \in I\} \subseteq L[x]$ である. 多項式 $f_1, \ldots, f_k \in K[t]$ に対して， $f_1, \ldots, f_k$ によって定義される代数多様体を $\mathbf{V}(f_1, \ldots, f_k)$ とする．すなわち, $\mathbf{V}(f_1, \ldots, f_k) := \{\bar{a} \in L^m \mid f_1(\bar{a}) = \cdots = f_k(\bar{a}) = 0\}$ である.

本稿では，代数構造的集合を $\mathbf{V}(f_1, \ldots, f_k) \setminus \mathbf{V}(g_1, \ldots, g_l) \subseteq L^m$ で表し，セル (cell) と呼ぶ. (ただし, $f_1, \ldots, f_k, g_1, \ldots, g_l \in K[t]$ である). また，$I \subset K[x]$ をイデアルとするとき $\sqrt{I}$ は I の根基とする.

定義 1 (包括的グレブナ基底系)
F を $K[t][x]$ の有限部分集合, $\mathbb{A}_1, \ldots, \mathbb{A}_l$ を L^m 上の代数構造的集合（セル）, $G_1, \ldots, G_l$ を $K[t][x]$ の有限部分集合とする. このとき，有限部分集合 $\mathcal{G} = \{(\mathbb{A}_1, G_1), \ldots, (\mathbb{A}_l, G_l)\}$ が $\cup_{i=1}^{l} \mathbb{A}_i$ 上で $\langle F \rangle$ の包括的グレブナ基底系 *(Comprehensive Gröbner System (CGS))* とは，次の条件を満たすときである.

- 任意の $i, j \in \{1, 2, \ldots, l\}$ において $\mathbb{A}_i \cap \mathbb{A}_j = \emptyset$ である．$(i \neq j)$
- 各 $1 \leq i \leq l$ に対して，任意の $\bar{a} \in \mathbb{A}_i$ と $g \in G_i$ において，$lt_x(g) = lt_x(\sigma_{\bar{a}}(g))$ かつ $\sigma_{\bar{a}}(G_i)$ が $L[x]$ 上のイデアル $\langle \sigma_{\bar{a}}(F) \rangle$ のグレブナ基底である.

また, 各 $(\mathbb{A}_i, G_i)$ を $\mathcal{G}$ の セグメントという．もし $\cup_{i=1}^{l} \mathbb{A}_i = L^m$ であれば, $\mathcal{G}$ を単に $\langle F \rangle$ の包括的グレブナ基底系という.

包括的グレブナ基底系の計算アルゴリズムは大きく分けて 4 つ存在する．(下線は本研究で使用したもの．)

- Weispfenning-Montes：S 多項式を計算する際，先頭項がゼロかゼロでないかで分けてグレブナ基底を計算する方法.
  1992 年 Weispfenning[18], 2002 年 Montes[7], 2003 年 Weispfenning[19], 2006 年 Montes[6], 2010 年 Montes[8]

- Alternative Comprehensive Gröbner Basis(ACGB) : von Neumann 正規環を係数環とする多項式環のイデアルのグレブナ基底を計算する方法.
  2001 年 Sato-Suzuki(DCGB)[15], 2003 年 Suzuki-Sato(ACGB)[16, 20],
  2003 年 Sato-Suzuki-Nabeshima(DCGB,ACGB-V)[13, 14], 2007 年 Kurata-Noro(DCGB)[5]
- Suzuki-Sato : ブロック項順序を用いて多項式環を係数環とする多項式環のイデアルのグレブナ基底を繰り返し計算する方法.
  2006 年 Suzuki-Sato[17], 2007 年 Nabeshima[9], 2010 年 Kapur-Sun-Wang[4], 2012 年 Nabeshima[10]
- Nabeshima : 有理関数体を係数とするグレブナ基底計算とイデアル商計算を繰り返し行う方法.
  2024 年 Nabeshima[11]

今回はイデアルの安定性理論を用いた Suzuki-Sato と Nabeshima のアルゴリズムが速いと言われているので，それらを融合する．

ここでは Suzuki-Sato アルゴリズムを改良した Kapur-Sun-Wang[4] の包括的グレブナ基底系計算アルゴリズムを紹介する．まず，使用する Minimal Dickson basis を定義する．

定義 2 (Minimal Dickson basis)
$pp(x)$ の項順序 $\prec$ を固定し，F, G を $K[t][x]$ の有限部分集合とする．以下を満たすならば $F \subset K[t][x]$ は G の *Minimal Dickson basis* であるという．

- F が G の部分集合である．
- 任意の $g \in G$ に対して $\mathrm{lt}_x(g)|\mathrm{lt}_x(f)$ となるような $f \in F$ が存在する．すなわち，$\langle \mathrm{lt}_x(F)\rangle = \langle \mathrm{lt}_x(G)\rangle$ である．
- 異なる任意の $f_1, f_2 \in F$ に対して，$\mathrm{lt}_x(f_1) \nmid \mathrm{lt}_x(f_2)$ かつ $\mathrm{lt}_x(f_2) \nmid \mathrm{lt}_x(f_1)$ である．

Kapur-Sun-Wang のアルゴリズムは次の定理に基づいている．

定理 3 (2010, Kapur-Sun-Wang)
$E \subset K[t], F \subset K[t][x]$ を有限部分集合とする．$pp(x)$ 上の項順序 $\prec_x$ に関する $\langle F \cup E\rangle$ のグレブナ基底を G とする．$G_1 = G \backslash G \cap \langle E\rangle$ とし，G_1 の *Minimal Dickson basis* を $G_2, h = lcm(lc_x(g)|g \in G_2)$ とする．このとき，任意の $\bar{a} \in \mathbf{V}(E)\backslash\mathbf{V}(h) \subset L^m$ において，$\sigma_{\bar{a}}(G_2)$ は $\langle \sigma_{\bar{a}}(F)\rangle$ の $\prec_x$ に関するグレブナ基底である．

定理 3 は $\langle F \cup E\rangle$ のグレブナ基底 G の計算がメイン計算である．次に，Nabeshima の包括的グレブナ基底系計算アルゴリズムについて述べる．Nabeshima のアルゴリズムは次の定理に基づいている．

定理 4 (2024, Nabeshima)
$E \subset K[t], F \subset K[t][x]$ を有限部分集合とする．$\langle E\rangle$ の $K[t]$ における極大独立集合を u とし，$K(u)[t\backslash u, x]$ で $t\backslash u \ll x$ に関するブロック項順序 $(\prec_{t\backslash u}, \prec_x)$ で $\langle F \cup E\rangle$ のグレブナ基底 G を計算する．$G_1 = G\backslash G \cap \langle E\rangle, G_1$ の *Minimal Dickson basis* を G_2 とする．イデアル商 $\langle F \cup E\rangle : \langle G_1\rangle$ の簡約グレブナ基底 S を $u \ll t\backslash u \ll x$ のブロック項順序で計算する．また，$h = lcm(lc_x(g)|g \in G_2)$ とし，$S \cap K[u]$ から 1 つ元 q をとる．このとき，任意の $\bar{a} \in \mathbf{V}(E)\backslash\mathbf{V}(h \cdot q) \subset L^m$ において，$\sigma_{\bar{a}}(G_2)$ は $\langle \sigma_{\bar{a}}(F)\rangle$ の $\prec_x$ に関するグレブナ基底である．

これは，$\langle F \cup E\rangle$ のグレブナ基底 G の計算と，イデアル商の簡約グレブナ基底 S の計算がメインの計算である．

次に，$\langle E\rangle \subset K[t]$ がゼロ次元の場合を考える．このとき，特別な計算法が存在する．

可換な環 R が可換な von Neumann 正規環であるとは，任意の $a \in R$ のとき，$a^2b = a$ となるような $b \in R$ が存在することである．

定理 5 (2003, Sato-Suzuki-Nabeshima)
$\langle E\rangle$ をゼロ次元, $F \subset K[t][x]$ とする．このとき，$K[t]/\sqrt{\langle E\rangle}$ は可換な *von Neumann* 正規環となるので，$\langle F\rangle$ を $(K[t]/\sqrt{\langle E\rangle})[x]$ 上のイデアルとみなし，$(K[t]/\sqrt{\langle E\rangle})[x]$ 上での $\langle F\rangle$ のグレブナ基底を G とする．このとき，任意の $\bar{a} \in \mathbf{V}(E)$ において，$\sigma_{\bar{a}}(G)$ は $\langle \sigma_{\bar{a}}(F)\rangle$ のグレブナ基底となる．

これは $\langle F\rangle$ のグレブナ基底を可換な von Neumann 正規環を係数環とする多項式環上で計算する方法であり，既に ISSAC 2007 倉田-野呂のプログラムが存在する．

定理 6 (1997, Kalkbrener)
$\langle E\rangle \subset K[t]$ をゼロ次元, $F \subset K[t][x]$ とし，$\sqrt{\langle E\rangle}$ の素イデアル分解を $\sqrt{\langle E\rangle} = P_1 \cap P_2 \cap \cdots \cap P_l$ とする．$\langle F\rangle \cup P_i$ のグレブナ基底 G' を，$K[t,x]$ において項順序 $t \ll x$ で計算し，$G = G' \backslash (G' \cap P_i)$ とする．このとき， 任意の $\bar{a} \in \mathbf{V}(P_i), \sigma_{\bar{a}}(G)$ は，$\langle \sigma_{\bar{a}}(F)\rangle$ のグレブナ基底となる．

$\sqrt{\langle E\rangle}$ の素イデアル分解は高速に計算可能であり，$\langle F\rangle \cup P_i$ のグレブナ基底を $K[t,x]$ で計算することで包括的グレブナ基底の 1 つのセグメントを得ることができる．

3　結果 1

前述した定理 3, 4 を組み合わせ，多コアな CPU を持つパソコンで計算を行うことで計算時間が速くならないかと考えた．また，ゼロ次元の場合，定理 3 と 4 は本質的に同じであるため，定理 3 を使用し，かつ，ゼロ次元特化型である定理 5, 6 の 3 つを使用する．

3.1　アルゴリズムの融合

$F = \{ax^2 - xy + y^2, bxy + y, ax^2 - y, (b+1)xy^2 + ax\} \subset \mathbb{C}[a,b][x,y]$ を，辞書式項順序 $y \prec x$ に関する包括的グレブナ基底系を考える．

今までは，1 つの戦略のみを使用していた．図 1 では定理 3 を使用し，セグメント $(\mathbb{C}^2 \backslash (\mathbf{V}(a(a+b^2+b)), \{(-a-b^2-b)y, -ax-b\})$ が得られた．得られたセグメントの条件から，図 1 のように $a+b^2+b=0$ と $a=0$ という条件にわけて再度計算を繰り返す．

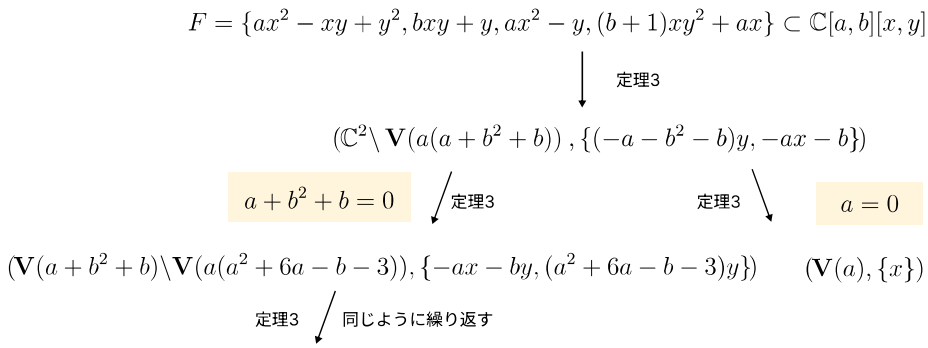

図 1: 今までのアルゴリズム

一方，本研究での提案手法は前述した定理 3, 4 を同時に走らせ，速く結果が得られた方を採用し，そのセグメントの条件を使用して次の計算に入る．しかし，計算速度は問題とアルゴリズムに依存するため，速いものから得られたセグメント条件を使用しても次の計算も速くなるとは限らない．そこで，図 2 に示し

たように 1 回目の計算のみどの戦略を使用するか選択できるように実装した.

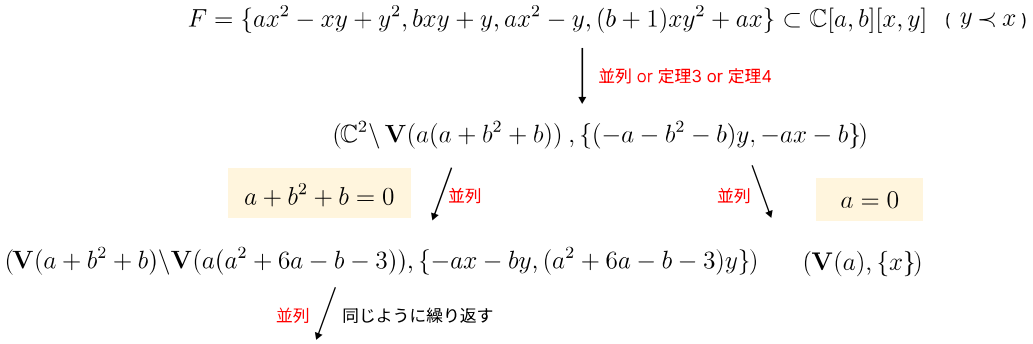

図 2: 今回の融合のアイデア

計算を繰り返していくうちにゼロ次元である場合が出てくることがある．そこでゼロ次元の場合は前述したゼロ次元特化版のアルゴリズム (定理 3, 5, 6) を使用する．

3.2 実装方法

Risa/Asir を使用し，図 3 のように実装した．入力は，多項式, パラメータ, 変数, 項順序, タイプの 5 つとした．タイプとは，1 回目の計算を選択するためのものであり，以下のようになっている．

Type=0 : 2 つの戦略を同時に走らせ，速い方を採用

Type=1 : 1 回目の計算は，定理 3(Kapur-Sun-Wang) のみを使用．2 回目以降は 2 つの戦略を並列.

Type=2 : 1 回目の計算は，定理 4(Nabeshima2024) のみを使用．2 回目以降は 2 つの戦略を並列.

図 3 は実装アルゴリズムである．本章では，戦略 1：定理 3, 戦略 2：定理 4, ゼロ次元戦略 1：定理 5, ゼロ次元戦略 2：定理 6 を意味する．1 つの計算が終わり，得られたセグメントからゼロ次元かゼロ次元でないかを判定し，ゼロ次元でなければ，定理 3, 4 を並列，ゼロ次元であれば，ゼロ次元戦略の定理 3, 5, 6 の 3 つを並列させる．最後まで計算が終わったら，それぞれの定理の使用回数，計算時間，包括的グレブナ基底系，セグメント数などを出力できるよう実装した.

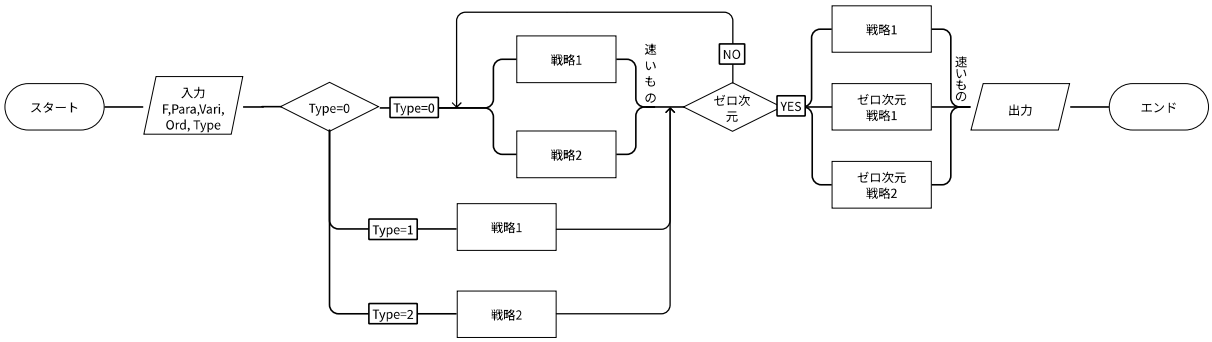

図 3: 実装アルゴリズム

3.3 並列処理の仕方

これらのアルゴリズムを Risa/Asir の OpenXM を用いて並列する．

$return$	整数
$func$	関数名
$host$	文字列または 0
$arg0\ldots$	任意 (引数)
$dir/command$	文字列
id	整数
$nlist$	数 (子プロセス識別子) のリスト

- ox_launch([host [,dir], command])
  host 上で command を起動し，このプロセスと通信を開始する．引数が 3 つの場合，host 上で dir にある ox_launch というサーバー起動用プログラムを立ち上げる．無引数の場合，host は 0，dir は get_rootdir() で返されるディレクトリ，command は同じディレクトリの ox_asir を意味する．

- ox_rpc(id, "func", arg0...)
  サーバーが ox_asir の場合にのみ使用可．それ以外の場合は，ox_cmo_rpc() を用いる．
  識別子 id のプロセスの関数を呼び出す．関数の計算終了を待たず，直ちに 0 を返す．

- ox_pop_local(id)
  識別子 id が返す値を取り出す．

- ox_push_cmd(id, command)
  識別子 id のプロセスに command を送信する．

- ox_select(nlist[,timeout])
  識別子リスト nlist のプロセスのうち既に出力を返している，読み出し可能なプロセスの識別子リストを返す．
  全てのプロセスが RUN 状態のとき，いずれかのプロセスの終了を待つ．但し，timeout が指定されている場合，timeout 秒だけ待つ．

- ox_get(id)
  プロセス識別子 id のプロセスからデータを受信する．ox_push_cmd と組み合わせて用いる．

- ox_reset(id)
  識別子 id のプロセスをリセットし，コマンドを受付状態にする．
  そのプロセスが既に書き出した，あるいは現在書き出し中のデータがある場合，それを全部読み出し，出力バッファを空にした時点で戻る．
  子プロセスが RUN 状態の場合でも，割り込みによって強制的に計算を終了させる．(return は 1)

- ox_shutdown(id)
  識別子 id に対応する遠隔プロセスを終了させ，通信を終了する．

また，計算時間は openXM の子プロセスは計測されないため，メインサーバーと並列用に子プロセスのサーバーそれぞれで計測し，使用したものだけを取得・合算する方法を採用した．

3.4　ベンチマーク

x, y, z を変数,a, b, c, d をパラメータとし，全次数逆辞書式順序として実行した結果から特徴的なものを抜粋した．

$$F1 = \{y^4z + xy^2 + x^3, y^2zax^3 + 4bx^3y^2, xy^3 + y^3 + ayz\}$$
$$F2 = \{x^5y^4 + by^2 + x^2y, x^4y + axy^2 + y, x^2y + ax^2\}$$
$$F3 = \{y^4z^4 + cy^2 + ax, y^5z + axy + 4bxy, x^2y^3z^4 + cy^3 + axz\}$$
$$F4 = \{y^3z^2 + xy + ax, y^5z + axyz + 2bx, xy^3z^4 + y^3 + axz\}$$
$$F5 = \{y^3z^2 + ax^2y^2 + x^3, y^3z^2 + ax^2yz + 2bx, xy^3z^4 + y^3 + az\}$$
$$F6 = \{x^3y + ax^4 + bxz + 5y, x^2y + ax^3z^2 + y^4, 4x^5z + xy^6 + y^2 + bx^5z^2\}$$
$$F7 = \{x^3y^6 + xy^3 + ay, x^2y + y^2 + xy, x^6y + by^2\}$$

単位は CPUsec. で使用 OS は，Windows 10 であり，計算機の性能は CPU: Intel(R) Core(TM) i9-C7900X CPU @ 3.30 GHz, RAM: 256GB である．

7 つのベンチマーク問題を，Algorithm 3(定理 3 のみ)，Algorithm 4(定理 4 のみ)，並列，1 回目定理 3 を使用しその後並列，1 回目定理 4 を使用しその後並列の 5 種類で実行した．

	定理3	定理4	並列	定理3→並列	定理4→並列
F1	0.011429	0.015625	(3, 0, 1, 0, 0) (4)	(3, 0, 1, 0, 0) (4)	(2, 1, 1, 0, 0) (4)
			0.011411	0.01855	0.03125
F2	0.28125	5.53125	(2, 1, 1, 0, 0) (7)	(2, 1, 1, 0, 0) (7)	(2, 1, 1, 0, 0) (7)
			0.03125	0.078125	0.09375
F3	49.1094	1422.78	(30, 1, 1, 0, 0) (32)	(30, 1, 1, 0, 0) (32)	(29, 1, 1, 0, 0) (32)
			17.2656	16.25	190.703
F4	over 4days	over 4days	(6, 0, 0, 0, 3) (14)	(6, 0, 0, 0, 3) (14)	(4, 1, 0, 0, 3) (13)
			12.1094	13.2344	888.859
F5	0.015625	3158.28	(3, 0, 1, 0, 0) (4)	(3, 0, 1, 0, 0) (4)	(2, 1, 1, 0, 0) (4)
			0.0625	0.140625	8309.31
F6	over 4days	98.75	(5, 0, 1, 0, 2) (10)	(5, 0, 2, 0, 1) (9)	(3, 1, 3, 0, 0) (7)
			1.59375	2.92188	2.73438
F7	5081.09	101.859	(3, 0, 1, 0, 0) (10)	(3, 0, 1, 0, 0) (10)	(1, 1, 1, 0, 0) (9)
			267.172	269.938	318.938

上記の表の並列箇所は次のような構成になっている．

(1,0,0,0,0) (1)
0.234567

上段の 2 つの括弧は使用回数 (定理 3, 4, ゼロ次元時の定理 3, 5, 6) とセグメント数を表している．右図の場合，定理 3 が 1 回，定理 4 が 0 回，ゼロ次元時の定理 3 が 0 回，定理 5 が 0 回，定理 6 が 0 回であり，得られたセグメントは 1 つ，実行時間が 0.234567 CPUsec. である．

この結果を考察する．

F1 全体的に速いので差は少ないが，ゼロ次元戦略を使用する並列が速い．

F2 全体的に速いので差は少ないが，定理 3 と 4 を使用し，ゼロ次元戦略を使用する並列が速い．

F3 定理 4 のみより，定理 3 のみの使用が速いが，並列すると複数種類使用することでより速くなる．

F4 1 つの戦略のみ使用する方法では 4 日以上たっても結果が出ないが，ゼロ次元戦略を使用することで 12 秒で結果が出る．

F5 定理 3 のみが速いとでるが，ほぼ誤差の範囲内と考えられる．

F6 並列が断然速い．

F7 定理 4 のみが速い．

以上より，今までの 1 つの戦略のみを使用する方法より速くなることが多い一方，F7 では必ずしも速くなるわけではないという結果が得られた．また，速さは問題とアルゴリズムに依存するが，どのアルゴリズムが速いかは実行するまでわからないことが多かったが，並列で処理することでその手間が省けるようになったのではないかと考える．また出力されたセグメント数に注目すると，定理 4 →並列が他より少ないことがわかる．

4　結果 2

追加研究として，パラメータの条件に着目した 2012 Nabeshima アルゴリズムとの融合を考える．

4.1　新たな安定性

定理 3 をパラメータ条件に注目したものが次の定理である．

定理 7 (Nabeshima2012)
$E \subset K[t], F \subset K[t][x]$ を有限部分集合とする．$K[t,x]$ 上で $t \ll x$ に関する項順序で $\langle F \cup E \rangle$ のグレブナ基底を G とする．*(x には $\succ$ を設定)* $G \backslash (G \cap \langle E \rangle)$ を G_1 とし，$\langle \mathrm{lt}_x(G_1) \rangle$ の極小基底を $M = \{m_1, \ldots, m_l\} \subset K[x]$ とする．$G_{m_i} = \{g \in G_1 | \mathrm{lt}_x(g) = m_i\} (1 \leq i \leq l)$ とする．このとき，$\bar{a} \in \mathbf{V}(E) \backslash (\mathbf{V}(\mathrm{lc}_x(G_{m_1})) \cup \mathbf{V}(\mathrm{lc}_x(G_{m_2})) \cup \cdots \cup \mathbf{V}(\mathrm{lc}_x(G_{m_l})))$ において，$\sigma_{\bar{a}}(G_{m_1} \cup G_{m_2} \cup \cdots \cup G_{m_l})$ は $\langle \sigma_{\bar{a}}(F) \rangle$ の $\succ$ に関するグレブナ基底である．

系 8
各 G_{m_i} から *1* つだけ元 g_i をとる．このとき，$\{g_1, \ldots, g_l\}$ は G_1の *Minimal Dickson basis* である．

例 1
$G = \{ax+b, bx+2\} \in \mathbb{C}[a,b][x]$ とする．$\langle lt_x(G) \rangle$ の極小基底は $\{x\}$ である．ここで定理 *3(KSW)* では G の *Minimal Dickson basis* は $\{ax+b\}$ であり，*Minimal Dickson basis* の先頭項の最小公倍元は a となる．すると得られるセグメントは $(\mathbb{C}^2 \backslash \mathbf{V}(a), \{ax+b\})$ となる．一方，定理 *7(Nabeshima 2012)* では $G_x = \{ax+b, bx+2\}$ となり，$\mathbf{V}(G_x) = \mathbf{V}(a,b)$ となるため，得られるセグメントは $(\mathbb{C}^2 \backslash \mathbf{V}(a,b), \{ax+b, bx+2\})$ となる．

つまり，$\mathbf{V}(a,b) \subset \mathbf{V}(a)$ より，$\mathbb{C}^2 \backslash \mathbf{V}(a) \subset \mathbb{C}^2 \backslash \mathbf{V}(a,b)$ となる．よって，パラメータ空間に着目すると定理 *7(Nabeshima2012)* が有利なことがわかる．ただし，グレブナ基底は極小でなく，どの先頭項が消えるかわからないため，グレブナ基底を用いて何かする際には向いていないことに注意する．あくまで次元の判定など先頭項だけが関わるものにはよく働く．

同様に定理 4(Nabeshima 2024) をパラメータに着目して書き換える．

定理 9 (Nabeshima2024)
$E \subset K[t], F \subset K[t][x]$ を有限部分集合とする．$\langle E \rangle$ の $K[t]$ における極大独立集合を u とし，$K(u)[t \backslash u, x]$ で $t \backslash u \ll x$ に関する項順序で $\langle F \cup E \rangle$ のグレブナ基底 G を計算する．(x には $\succ$ を設定する) $G \backslash (G \cap \langle E \rangle)$ を G_1 とし，イデアル商 $\langle F \cup E \rangle : \langle G \rangle$ の簡約グレブナ基底 S を $u \ll t \backslash u \ll x$ の項順序で計算する．$\langle \mathrm{lt}_x(G_1) \rangle$ の極小基底を $M = \{m_1, \ldots, m_l\} \subset K[x]$ とする．$G_{m_i} = \{g \in G_1 | \mathrm{lt}_x(g) = m_i\} (1 \leq i \leq l)$ とすると，このとき，任意の $\bar{a} \in \mathbf{V}(E) \backslash (\mathbf{V}(\mathrm{lc}_x(G_{m_1})) \cup \mathbf{V}(\mathrm{lc}_x(G_{m_2})) \cup \cdots \cup \mathbf{V}(\mathrm{lc}_x(G_{m_l})) \cup \mathbf{V}(S \cap K[u]))$ において $\sigma_{\bar{a}}(G_1)$ は $\langle \sigma_{\bar{a}}(F) \rangle$ の $\succ$ に関するグレブナ基底である．

メインの計算は定理 4 と変わらず，$\langle F \cup E \rangle$ のグレブナ基底計算とイデアル商の簡約グレブナ基底 S の計算である．

4.2 アルゴリズムの融合

アルゴリズムの融合アイデアは前述の融合アイデアと変わらない．定理 7 と定理 9 の 2 つの戦略を並列で実行し，ゼロ次元のときには定理 7 と定理 5, 6 の 3 つを並列実行する．
第 4 章 図 3 の実装アルゴリズムは，戦略 1 : 定理 7，戦略 2 : 定理 9，ゼロ次元戦略 1 : 定理 5，ゼロ次元戦略 2 : 定理 6 を意味する．結果 1 と同じ例題，$F = \{ax^2 - xy + y^2, bxy + y, ax^2 - y, (b+1)xy^2 + ax\} \subset \mathbb{C}[a, b][x, y]$ を，辞書式項順序 $y \prec x$ に関する包括的グレブナ基底系を考える．はじめに，定理 7 と 9 を並列すると，セグメント $\mathbb{C}^2 \backslash (\mathbf{V}(a) \cup \mathbf{V}(a + b^2 + b, (b-2)a + 1, a^2 + 6a - b - 3)), G_x \cup G_y$ が得られた．結果 2 では得られたセグメントの条件を 1 度に複数計算が可能である．

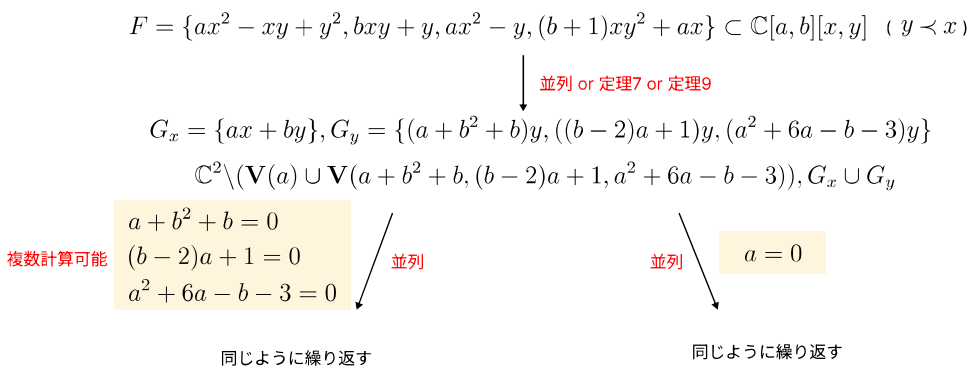

図 4: 追加研究のアイデア

4.3 追加研究のベンチマーク

問題は結果 1 と同様のものを使用する．
7 つのベンチマーク問題を，定理 7 のみ，定理 9 のみ，並列，1 回目定理 7 を使用しその後並列，1 回目定理 9 を使用しその後並列の 5 種類で実行した．x, y, z を変数,a, b, c, d をパラメータとし，全次数逆辞書式順序として実行した結果から特徴的なものを抜粋した．

結果 1 では並列すると速くなるのかということに着目したが，ここではパラメータ条件に着目した 2012 Nabeshima アルゴリズムについてみるため，結果 1 と結果 2 の並列処理の実行時間のみを比較する．使用回数やセグメントに関しては付録に記載する．

	結果1			結果2		
	並列	定理3 ›並列	定理4→並列	並列	定理7→並列	定理9 ›並列
F1	0.011411	0.01855	0.03125	0.016943	0.02419	0.086021
F2	0.03125	0.078125	0.09375	0.184191	0.199741	0.192489
F3	17.2656	16.25	190.703	1.88718	1.76829	225.698
F4	12.1094	13.2344	888.859	13.4438	15.0117	136.482
F5	0.0625	0.140625	8309.31	0.149892	0.210692	1018.93
F6	1.59375	2.92188	2.73438	2.38575	2.93486	3.70532
F7	267.172	269.938	318.938	1.15073	1.33651	1.62618

全体的に結果 1 の並列が速い傾向があるように見えるが，特徴的なものがある．F4, F5 は，どちらも並列が速いが，結果 1 の定理 4 →並列と結果 2 の定理 9 →並列を比べると結果 2 の方が速いことがわかる．結果 2 は結果 1 をパラメータの条件に注目し改良を行なったものであり，複数条件が出た際にそれらを複数同時に計算できることが強みである．そこで，この 2 つの実行途中で得られるセグメントの条件を出力してみたところ，先頭係数のリスト内部が分割されているものが一括で計算できるものであり，結果 1 より速くなる可能性があるものである．F4,F5 はそれが顕著に表れており，実行時間が速くなったのではないかと考えられる．

5　まとめ

計算実行時間は問題とアルゴリズムに依存するため，複数アルゴリズムを並列し速く出力されたものを採用する方法は有効であるものが多い．しかし，速く出力されたセグメントが次の計算で必ずしも速くならないことも明らかになった．また，出力されるセグメントの数にも違いがみられた．

これまで 2 通りの戦略のみの計算法だったが，並列で組み合わせることで更なる計算方法・実装が確立された．

参　考　文　献

[1] Becker, T., Weispfenning, V., *Gröbner Bases*, GTM 141, Springer, (1993)

[2] Cox, D., Little, J., O ’ Shea, D., *Ideals, Varieties, and Algorithms*, Springer, 2nd edition, 1997.

[3] Kalkbrener, M., On the stability of Gröbner basis under specializations, *J. Symbolic Compu.*, 24, 51-58, 1997.

[4] Kapur, D., Sun, Y., and Wang, D., A new algorithm for computing comprehensive Gröbner systems. *Proc. ISSAC 2010*, 29-36, ACM, (2010)

[5] Kurata, Y., Noro, M., Computation of discrete comprehensive Gröbner bases using modular dynamic evaluation. *Proc. ISSAC 2007*, 243-250, ACM, (2007)

[6] Manubens, M., Montes, A., Improving DISPGB algorithm using the discriminant ideal, *J. Symb. Comp.*, 41, 1245-1263, (2006).

[7] Montes, A., A new algorithm for discussing Gröbner bases with parameters, *J. Symb. Comp.*, 33, 183-208, (2002)

[8] Montes, A.,Wibmer, M., Gröbner bases for polynomial systems with parameters, *J. Symb. Comp.*, 45, 1391-1425, (2010)

[9] Nabeshima, K., A speed-up of the Algorithm for computing comprehensive Gröbner systems, *Proc. ISSAC 2007*, 299-306, ACM, (2007)

[10] Nabeshima, K., Stability conditions of monomial bases and conprehensive Gröbner systems, *Proc. CASC2012, Lecture Notes in Computer Science*, 7442, 248-259, Springer, (2012)

[11] Nabeshima, K., Generic Gröbner basis of a parametric ideal and its application to a comprehensive Gröbner system, *AAECC*, vol.35, 55-70, (2024)

[12] Noro, M. and Takeshima, T., Risa/Asir - a computer algebra system. *Proc. ISSAC 1992*, 387-396, ACM(1992)

[13] Sato, Y., Suzuki, A., and Nabeshima, K., ACGB on varieties, *Proc., CASC 2003*, 313-318, Universität München, (2003)

[14] Sato, Y., Suzuki, A., and Nabeshima, K., Discrete comprehensive Gröbner bases II, Computer Mathmatics Ⅲ, *Lecture Notes Series on Computing*, 240-247, World Scientific (2003)

[15] Suzuki, A., Sato, Y., Discrete comprehensive Gröbner bases. *Proc. ISSAC 2001*, 292-296, ACM, (2001)

[16] Suzuki, A., Sato, Y., An alternative approach to comprehensive Gröbner bases, *J. Symb. Comp.*, 36, 649-667, (2003)

[17] Suzuki, A., Sato, Y., A simple algorithm to compute comprehensive Gröbner bases using Gröbner bases, *Proc. ISSAC 2006*, 326-331, ACM, (2006)

[18] Weisphenning, V., Comprehensive Gröbner bases, *J. Symb. Comp.*, 14, 1-29, (1992)

[19] Weisphenning, V., Canonical comprehensive Gröbner bases, *J. Symb. Comp.*, 36, 669-683, (2003)

[20] Weisphenning, V., Comprehensive Gröbner bases and regular rings, *J. Symb. Comp.*, 41, 285-296, (2006)