数式処理的手法に基づく過少代数制約問題の数値解法

沢田浩之†

過少代数制約問題とは，決定すべき変数に対して等式制約が不足しているために解が一意に決定しない問題のことを指し，設計問題やレイアウト問題などはその典型的な例である．過少代数制約問題を解くためには，通常，問題解決にあたる作業者の知識や勘に基づいて変数のうちのいくつかに適当な値を与え，得られた結果を評価・修正することによって最終的な解を得るという方法が用いられる．そのため，存在する解を見落とす可能性や，存在しない解を探し求めることに労力を費す可能性は免れない．それらの可能性を排除するため，本論文では，過少代数制約問題を制約条件付最小値問題に帰着させることによってその数値解を求める方法を提案する．この方法は，与えられた問題に実数解が存在する場合にはその数値解を求めることが可能であり，したがって，数値解が得られない場合には，実数解が存在しないと見なしうる．本手法は，実際の設計問題やレイアウト問題解決において有用な情報を提供するツールとして用いることができるものと考えられる．

Numerical Solution of Under-constrained Algebraic Systems Based on Symbolic Computation

HIROYUKI SAWADA†

It is well known that a design problem or a layout problem can be formulated as an under-constrained algebraic problem. An under-constrained algebraic problem is defined as a problem in which solutions cannot be uniquely obtained because there are not enough equality constraints. Conventionally, an under-constrained algebraic problem is solved in the following way. Firstly, by assigning some concrete values to some variables based on human worker's knowledge or intuition, a temporary solution is made. After that, it is checked and modified. Then the solution to the given problem is finally obtained. Therefore, there are possibilities of looking over existing solutions and wasting time to search a solution which does not actually exist. In order to omit those possibilities, this paper proposes a new method of obtaining numerical solutions to an under-constrained algebraic problem by making the given problem result in a constrained minimization problem. This method can compute a numerical solution if there exists a real solution to the given problem. Thus, if no numerical solution is obtained, it can be concluded that there exists no real solution. This method is considered to be a tool which gives useful information for solving a design or layout problem.

1. はじめに

代数制約問題を解く場合には，通常，等式制約に基づいて求められた有限個数の解の中から，不等式制約を満足するものを選択するという方法が用いられる．過少代数制約問題とは，決定すべき変数に対して等式制約が不足しているために解が有限個とはならない問題のことを指す．設計問題やレイアウト問題などはその典型的な例といえる．たとえば，レイアウト問題における制約は，「物体Xと物体Yは定められた範囲内になくてはならない」，「物体Xと物体Yは隣接している」といった形で与えられ，これらの制約条件は代数不等式および代数方程式として表現されるが，物体の位置を具体的に定めるためには条件が不足している場合が多い．一般にこのような過少代数制約問題は解決の見通しが悪い．

過少代数制約問題を解くためには，従来，以下のような方法が多く用いられている．

(1) まず，決定すべき変数のうちのいくつかに適当な値を与えることによって，元の問題を解が有限個存在するような問題に帰着させる．

(2) 次に有限個の解を求め，その中から不等式制約を満足するものを選択する．

しかしながら，この方法には重大な欠点がある．それは，解が求められなかった場合，それが以下の2つ

† 機械技術研究所
Mechanical Engineering Laboratory

の理由のうちのいずれによるものかを特定できないということである．

- あらかじめ仮定した変数の値が不適当であった．
- 元の問題にそもそも解が存在しなかった．

したがって，このような方法を用いた場合には，存在する解を見落とす可能性や存在するはずのない解を探し求めることに労力を費やしてしまう可能性は免れない．

過少代数制約問題の実数解の有無を判定する方法としては，Quantifier Elimination（QE）に関する研究[3]がよく知られている．だが，QEは実数解の有無の判定を正確に行うことはできるものの，具体的な数値解を得るにはいたらない．

本論文では，与えられた過少代数制約を適切に評価し，実数解が存在する場合にはその数値解を得ることを目的として，過少代数制約問題を制約条件付最小値問題に帰着させ，これを解くことによって解を得る方法を提案する．本論文で提案する方法では，まず，与えられた過少代数制約によって空間内に定義される領域 A を考える．その際，スラック変数を導入することによって，領域 A を境界の閉じた領域としておく．過少代数制約問題の解とは，その領域 A の中に存在する点にほかならない．ここで，最小値を持つ目的関数を導入する．領域 A が空集合でなければ，この目的関数は領域 A 内で最小値を持つ．最小値が存在しなければ，それは領域 A が空集合であることを意味する．このようにして過少代数制約問題は制約条件付最小値問題に帰着され，与えられた制約条件のもとにおける目的関数の最小値を計算することによって，元の問題の解が求められる．

制約条件付最小値問題の解法についてはこれまでに多くの研究がなされており，ペナルティ法やバリア法などの数値解法が考案されている[7]．しかしながら，これら従来手法の多くは反復法に基づいた数値計算法であるため，解が収束しなかった場合，それが初期条件や収束条件が適切でなかったためなのか，元の問題に解が存在しなかったためなのかが明確ではない．したがって，本論文の目的である過少代数制約問題の評価にこれらの数値計算法を用いることは適切ではない．また，初期条件や収束条件の影響を受けることのない代数的手法としてはQEの概念に基づいた方法[11),12)]が提案されているが，その取り扱うことのできる制約条件は各変数の次数が2次以下のものに限定されており，一般性に欠ける．

そこで本論文では，代数制約一般について適用可能な，数式処理手法に基づいた制約条件付最小値問題の解法を新たに提案し，これを用いることにより過少制約問題の評価を行う．本論文で提案する手法により，従来試行錯誤によって求めるしかなかった過少代数制約問題の実数解を確実に得られるようになるとともに，実数解が存在しない場合には，そのことを明確に示すことが可能となる．

本論文の構成は以下のとおりである．まず，2章において，問題の設定を行ったのち，過少制約問題が制約充足最小値問題に帰着されることを示す．次に3章では，本論文で新たに提案する数式処理手法に基づいた制約条件付最小値問題の解法について述べる．4章では，本手法のインプリメントについて述べた後，これを応用した例を，歯車軸のレイアウト問題を用いて示す．

なお本論文では，制約として代数制約，解として実数解のみを対象としている．

2. 問題の設定

本章では，問題の設定を行うとともに，過少代数制約問題が制約条件付最小値問題に帰着されることを示す．

与えられた代数制約を式(1)とする．

$$\begin{aligned} &f_1(\boldsymbol{x}) = 0, \ldots, f_p(\boldsymbol{x}) = 0, \\ &g_1(\boldsymbol{x}) \neq 0, \ldots, g_q(\boldsymbol{x}) \neq 0, \\ &h_1(\boldsymbol{x}) \geq 0, \ldots, h_r(\boldsymbol{x}) \geq 0. \end{aligned} \tag{1}$$

ここで，$\boldsymbol{x} = (x_1, \ldots, x_n)$ は n 次元実数空間における変数，$f_i(\boldsymbol{x})$，$g_j(\boldsymbol{x})$，$h_k(\boldsymbol{x})$ は実数係数の多項式である．また，境界を含まない不等式制約 $e(\boldsymbol{x}) > 0$（$e(\boldsymbol{x})$ は実数係数の多項式）が与えられた場合，それは $e(\boldsymbol{x}) \neq 0 \wedge e(\boldsymbol{x}) \geq 0$ として扱われるものとする．

本論文で扱う問題は次のように定義される．

問題

(1) 式(1)に解が存在する場合，解の存在を示すために，それらの解の中から少なくとも1つを例として求める．

(2) 式(1)に解が存在しない場合，解が存在しないことを示す． □

この問題を解くために，本論文では過少代数制約問題(1)を制約条件付最小値問題に帰着させ，これを解くことにより解を得る方法を提案する．

スラック変数 $\boldsymbol{s} = (s_1, \ldots, s_q)$ および $\boldsymbol{t} = (t_1, \ldots, t_r)$ を導入することにより，式(1)を次式(2)に置き換える．

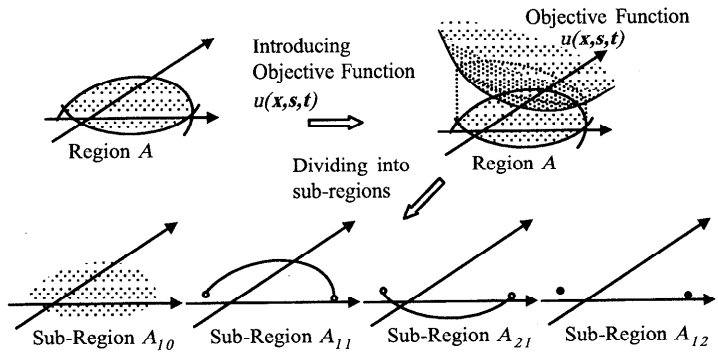


図1 制約条件付最小値問題への帰着
Fig. 1 Transformation to a minimization problem with constraints.

$$
\begin{aligned}
&f_1(\boldsymbol{x}) = 0, \ldots, f_p(\boldsymbol{x}) = 0,\\
&g_1(\boldsymbol{x}) \cdot s_1 = 1, \ldots, g_q(\boldsymbol{x}) \cdot s_q = 1,\\
&h_1(\boldsymbol{x}) = t_1, \ldots, h_r(\boldsymbol{x}) = t_r,\\
&t_1 \geq 0, \ldots, t_r \geq 0.
\end{aligned} \tag{2}
$$

$(n+q+r)$ 次元実数空間において，式 (2) で表される領域を A とする．

$$
\begin{aligned}
A = \{(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t}) | &f_1(\boldsymbol{x}) = 0, \ldots, f_p(\boldsymbol{x}) = 0,\\
&g_1(\boldsymbol{x}) \cdot s_1 = 1, \ldots, g_q(\boldsymbol{x}) \cdot s_q = 1,\\
&h_1(\boldsymbol{x}) = t_1, \ldots, h_r(\boldsymbol{x}) = t_r,\\
&t_1 \geq 0, \ldots, t_r \geq 0\}.
\end{aligned} \tag{3}
$$

ここで，以下の性質を持つ目的関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ を導入する．ただし，$u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ の定義域は $(n+q+r)$ 次元実数空間，$u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ の値は実数値である．

目的関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$

(1) 関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ は，$(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ 空間において連続である．

(2) 関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ は，$(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ 空間において最小値を持つ．

(3) $x_1, \ldots, x_n$, $s_1, \ldots, s_q$, $t_1, \ldots, t_r$ のうちの少なくとも 1 つがプラス無限大もしくはマイナス無限大となるとき，関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ はプラス無限大に発散する． □

領域 A の境界はすべて A に含まれているので，A が空集合でなければ，関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ は領域 A において最小値を持つ．したがって，領域 A における関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ の最小値を計算することにより，式 (2) の解がその最小点の座標値として求められる（図1）．このようにして，過少代数制約問題 (2) は，制約条件付最小値問題に帰着される．関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ が領域 A において最小値を持たなければ，それは A が空集合であることを意味し，過少代数制約問題 (2) に実数解は存在しない．

3. 制約条件付最小値問題

本章では，前章において定義された制約条件付最小値問題の解法について述べる．前章で導入した目的関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ にさらに任意の点において微分可能であるという条件を付加すると，その最小点は極値点を計算することによって求められる．本論文で提案する方法は，領域 A における関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ の極値点を計算するものであり，以下の 3 つの段階からなる．

(1) 領域 A の部分領域への分割．

(2) 各部分領域における極値点の計算．

(3) 極値点が無限個存在する場合の再帰的計算．

以下，3.1 節，3.2 節，3.3 節において，それぞれの段階について述べる．

なお，関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ の領域 A における最小点を得るためには求られた各極値点における関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ の値を比較する必要があるが，本論文ではそのような比較は省かれている．これは以下の理由による．本論文の最終的な目的は，与えられた過少代数制約問題を解くことにある．制約条件付最小値問題を解くことはそのための手段であり，関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ は 2 章の「目的関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$」で述べた 3 つの条件を満足しさえすれば任意に選ぶことができる．したがって，与えられた過少代数制約問題の実数解を少なくとも 1 つ求めるという目的に関していえば，関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ の極値点を少なくとも 1 つ求められれば十分であり，複数個の極値点の中から最小点を選び出すことには意味がないからである．

3.1 領域 A の部分領域への分割

極値点の座標は，関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ の $(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ に関する導関数を導き，その値が 0 となる点を計算することによって求められる．その際，導関数の値が不連続と

なる部分が存在すると極値点が正しく計算されない．したがって，領域を導関数が連続な部分領域に分割し，各部分領域において極値点の計算を行う必要がある．本論文で対象とする問題，式(2)の場合，式(2)の第4式によって与えられる領域 A の境界部において導関数の値が不連続となるので，極値点の計算に先立ち，領域 A を次式(4)のように分割する（図1）．

$$\begin{aligned} A &= \bigcup A_{ij}, \\ A_{ij} &= A \cap \{(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t}) | \\ &\qquad t_{a_j(i,1)} = \ldots = t_{a_j(i,j)} = 0, \\ &\qquad t_{a_j(i,j+1)} > 0, \ldots, t_{a_j(i,r)} > 0\}. \end{aligned} \tag{4}$$

ここで，$\{a_j(i,1),\ldots,a_j(i,j)\}$ は，1から r までの整数のうちの j 個の整数からなる組合せであり，全部で ${}_rC_j$ 通りの組合せが存在する．i はこれらの組合せを区別するためにつけられる添字であり，1から ${}_rC_j$ までの整数値をとる．たとえば，$r=3$ の場合，部分領域 A_{ij} と添字 $a_j(i,1),\ldots,a_j(i,r)$，およびスラック変数 t_k に関する方程式ならびに不等式との対応関係は以下のようになる．

A_{10}： $a_0(1,1)=1,\ a_0(1,2)=2,\ a_0(1,3)=3$
$(t_1>0,\ t_2>0,\ t_3>0)$.
A_{11}： $a_1(1,1)=1,\ a_1(1,2)=2,\ a_1(1,3)=3$
$(t_1=0,\ t_2>0,\ t_3>0)$.
A_{21}： $a_1(2,1)=2,\ a_1(2,2)=1,\ a_1(2,3)=3$
$(t_2=0,\ t_1>0,\ t_3>0)$.
A_{31}： $a_1(3,1)=3,\ a_1(3,2)=1,\ a_1(3,3)=2$
$(t_3=0,\ t_1>0,\ t_2>0)$.
A_{12}： $a_2(1,1)=1,\ a_2(1,2)=2,\ a_2(1,3)=3$
$(t_1=0,\ t_2=0,\ t_3>0)$.
A_{22}： $a_2(2,1)=1,\ a_2(2,2)=3,\ a_2(2,3)=2$
$(t_1=0,\ t_3=0,\ t_2>0)$.
A_{32}： $a_2(3,1)=2,\ a_2(3,2)=3,\ a_2(3,3)=1$
$(t_2=0,\ t_3=0,\ t_1>0)$.
A_{13}： $a_3(1,1)=1,\ a_3(1,2)=2,\ a_3(1,3)=3$
$(t_1=0,\ t_2=0,\ t_3=0)$.

極値点の計算は各部分領域 A_{ij} において行われる．部分領域 A_{ij} が有限個数の点，すなわち領域 A の頂点である場合にはそれを直接計算すればよい．そうでない場合には，次節以降に述べる方法に従って極値点の計算を行う．なお，部分領域 A_{ij} が有限個数の点であるか否かは，以下の手順で判定される．

部分領域 A_{ij} が有限個数の点であるか否かの判定法

(1) A_{ij} の定義式(4)から不等式を取り除くことにより次式(5)を得る．

$$\begin{aligned} &f_1(\boldsymbol{x})=0,\ldots,f_p(\boldsymbol{x})=0, \\ &g_1(\boldsymbol{x})\cdot s_1=1,\ldots,g_q(\boldsymbol{x})\cdot s_q=1, \\ &h_1(\boldsymbol{x})=t_1,\ldots,h_r(\boldsymbol{x})=t_r, \\ &t_{a_j(i,1)}=0,\ldots,t_{a_j(i,j)}=0. \end{aligned} \tag{5}$$

(2) 式(5)から，そのグレブナ基底[2] G を計算し，G に含まれる要素の頭項[8]からなる集合 H を作成する．

$$H=\{Ht(poly)|poly\in G\}.$$

ただし，$Ht(poly)$ は $poly$ の頭項を示す．

(3) 自然数全体の集合を $\boldsymbol{N}$ とし，式(5)に現れる変数全体の集合を Z とする（$Z=\{x_1,\ldots,x_n,s_1,\ldots,s_q,t_1,\ldots,t_r\}$）．次式が成立すれば，部分領域 A_{ij} は有限個数の点からなる．これが成立しなければ，部分領域 A_{ij} は有限個数の点ではない．

$$\forall(z\in Z)\exists(k\in\boldsymbol{N})\{z^k\in H\}. \qquad \square$$

3.2 部分領域 A_{ij} における極値点の計算

本論文で提案する手法では，各部分領域 A_{ij} における極値点の計算に Lagrange 乗数法[7]を用いる．Lagrange 乗数法では関数 $u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ の停留点が計算される．停留点とは，与えられた制約条件のもとで関数 $u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ の1階微分が0となる点を指す．通常，極値点は計算された停留点の中に含まれている．しかしながら，例外的に，停留点の中に含まれない極値点が存在する場合がある．Lagrange 乗数法は，陰関数定理の成立を前提とした解法である．したがって，陰関数定理が成立しない点は特異点となり，たとえその点が極値点であっても停留点として計算されない場合がある．

したがって，各部分領域 A_{ij} における極値点の計算は，以下の2種類の計算を行うことにより求められる．

(1) 部分領域 A_{ij} における関数 $u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ の停留点の計算．

(2) 部分領域 A_{ij} の特異点の計算．

3.2.1 項および 3.2.2 項において，それぞれ停留点の計算および特異点の計算について述べる．

3.2.1 停留点の計算

停留点の計算は以下の手順で行われる．

停留点の計算手順

(1) 式(5)で与えられる等式制約のもとにおいて，関数 $u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ の停留点を計算する．

(2) 計算された停留点の中から，部分領域 A_{ij} に関する不等式制約 $t_{a_j(i,j+1)}>0,\ \ldots,\ t_{a_j(i,r)}>0$ を満足する点を求める． $\square$

等式制約 (5) のもとにおける停留点の計算には，原理的には Lagrange 乗数法を用いる．本論文で提案する方法では，目的関数 $u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ に実数係数の多項式を使い，グレブナ基底計算を導入することにより Lagrange 乗数を含む項を取り除き，停留点の軌跡を表す連立方程式を，Lagrange 乗数を含まない形で求められるようになっている．

Lagrange 関数 $L(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t},\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\nu},\boldsymbol{\eta})$ は次式 (6) により与えられる．

$$\begin{aligned} L(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t},\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\nu},\boldsymbol{\eta}) &= u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t}) \\ &+ \sum_{k=1}^{p} \lambda_k f_k(\boldsymbol{x}) + \sum_{k=1}^{q} \mu_k (g_k(\boldsymbol{x})\cdot s_k - 1) \\ &+ \sum_{k=1}^{r} \nu_k (h_k(\boldsymbol{x}) - t_k) + \sum_{k=1}^{j} \eta_k t_{a_j(i,k)}. \end{aligned} \quad (6)$$

ただし，$\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, $\boldsymbol{\nu}$, $\boldsymbol{\eta}$ は Lagrange 乗数である．

停留点は，式 (5) と式 (7) を同時に満足する $(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ として与えられる．

$$\frac{\partial L}{\partial z_k} = 0 \ (k = 1,\ldots,n+q+r),$$

$$z_k = \begin{cases} x_k \ (k = 1,\ldots,n), \\ s_{k-n} \ (k = n+1,\ldots,n+q), \\ t_{k-n-q} \\ \quad (k = n+q+1,\ldots,n+q+r). \end{cases} \quad (7)$$

ここで，Lagrange 乗数を取り除き，停留点の軌跡 S を得る方法について述べる．式 (5) と式 (7) により生成される $\boldsymbol{R}[\boldsymbol{x},\boldsymbol{s},\boldsymbol{t},\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\nu},\boldsymbol{\eta}]$ におけるイデアル[10] を I とする．ただし，$\boldsymbol{R}[\boldsymbol{x},\boldsymbol{s},\boldsymbol{t},\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\nu},\boldsymbol{\eta}]$ は実数係数の多項式環である．停留点の軌跡は，イデアル I に含まれる要素のうち，$(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ のみからなる要素の集合として与えられる．したがって，イデアル I の部分イデアル $I' = I \cap \boldsymbol{R}[\boldsymbol{x},\boldsymbol{s},\boldsymbol{t}]$ の基底を求めれば，それが停留点の軌跡 S を与える連立方程式となる．その導出手順は以下のとおりである．

停留点の軌跡 S の導出手順

(1) 式 (5) と式 (7) からなる連立方程式のグレブナ基底 G を計算する．グレブナ基底 G の計算は，$\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, $\boldsymbol{\nu}$, $\boldsymbol{\eta} \succ \boldsymbol{x}$, $\boldsymbol{s}$, $\boldsymbol{t}$ に関する辞書式順序[2] で行われる．なお，全体の項順序には，辞書式順序と全次数逆辞書式順序による混在型順序を用いている（付録 A.1 参照）．これは，全体の項順序に辞書式順序を用いるよりも混在型順序を用いた方が，一般に計算効率が良いと考えられるからである．

(2) グレブナ基底 G の中から，Lagrange 乗数 $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, $\boldsymbol{\nu}$, $\boldsymbol{\eta}$ を含む要素を取り除く．このようにして得られた G の部分集合が，停留点の軌跡 S を与える連立方程式となる． □

停留点の座標 $(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ のみに関する連立方程式を導くことの利点は，停留点が無限個存在する場合，その解析が容易になるという点にある．停留点が無限個存在する場合，それらの中に部分領域 A_{ij} に関する不等式制約 $t_{a_j(i,j+1)} > 0, \ldots, t_{a_j(i,r)} > 0$ を満足する点が存在するか否かを調べるため，さらに計算を進めなくてはならない．その際，Lagrange 乗数を残しておいたままだと，それらを取り除いた場合に比べ，取り扱わなくてはならない変数が増えることになり，解析の見通しが悪くなるばかりでなく，計算効率上も不利になると考えられる．

停留点が無限個存在する場合の計算方法については，3.3 節で述べる．

3.2.2 特異点の計算

部分領域 A_{ij} の中で，$(p+q+r+j)$ 行 $(n+q+r)$ 列のヤコビ行列 $J(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ の階数が等式制約の数 $(p+q+r+j)$ よりも小さくなる点では陰関数定理が成立しない．そのような点は特異点となり，たとえ極値点であっても 3.2.1 項で述べた方法では停留点として検出されない場合があるため，別途計算を行う必要がある．

$$J(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t}) = \left(\frac{\partial W_k}{\partial z_l}\right)_{k=1,\ldots,p+q+r+j;\ l=1,\ldots,n+q+r},$$

$$W_k = \begin{cases} f_k(\boldsymbol{x}) \ (k = 1,\ldots,p), \\ g_{k-p}(\boldsymbol{x})\cdot s_{k-p} - 1 \\ \quad (k = p+1,\ldots,p+q), \\ h_{k-p-q}(\boldsymbol{x}) - t_{k-p-q} \\ \quad (k = p+q+1,\ldots,p+q+r), \\ t_{a_j(i,k-p-q-r)} \\ \quad (k = p+q+r+1,\ldots, \\ \qquad p+q+r+j), \end{cases} \quad (8)$$

$$z_l = \begin{cases} x_l \ (l = 1,\ldots,n), \\ s_{l-n} \ (l = n+1,\ldots,n+q), \\ t_{l-n-q} \\ \quad (l = n+q+1,\ldots,n+q+r). \end{cases}$$

特異点の計算は，停留点の計算と同様，以下の手順で行われる．

特異点の計算手順

(1) 式 (5) で与えられる等式制約における特異点を計算する．

(2) 計算された特異点の中から，部分領域 A_{ij} に関する不等式制約 $t_{a_j(i,j+1)} > 0, \ldots, t_{a_j(i,r)} > 0$ を満足する点を求める． □

ヤコビ行列 $J(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ の階数が $(p+q+r+j)$ よりも小さいということは，$J(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ の行ベクトルが1次従属であるということにほかならない．すなわち，特異点においては次式 (9) が成立する．

$$\exists c_1,\ldots,c_{p+q+r+j},\ \neg(c_1=\ldots=c_{p+q+r+j}=0)$$
$$c_1\begin{Bmatrix}\frac{\partial W_1}{\partial z_1}\\ \vdots \\ \frac{\partial W_1}{\partial z_{n+q+r}}\end{Bmatrix}+ \qquad (9)$$
$$\cdots+c_{p+q+r+j}\begin{Bmatrix}\frac{\partial W_{p+q+r+j}}{\partial z_1}\\ \vdots \\ \frac{\partial W_{p+q+r+j}}{\partial z_{n+q+r}}\end{Bmatrix}=\begin{Bmatrix}0\\ \vdots \\ 0\end{Bmatrix}.$$

したがって，式 (5) と式 (9) からなる連立方程式に対して，$c_1=1,\ldots,c_{p+q+r+j}=1$ なる制約をそれぞれ1つずつ追加することにより，前項で停留点の軌跡を求めたものと同様に以下の手順で，各 $c_k\neq 0$ に対応する特異点の軌跡 T_k を求めることができる．

特異点の軌跡 T_k の導出手順

(1) 式 (5)，式 (9) および $c_k=1$ $(1\leq k\leq p+q+r+j)$ からなる連立方程式のグレブナ基底 G を計算する．グレブナ基底 G の計算は，$c_1,\ldots,c_{p+q+r+j}\succ \boldsymbol{x},\boldsymbol{s},\boldsymbol{t}$ に関する辞書式順序で行われる．なお，全体の項順序には，辞書式順序と全次数逆辞書式順序による混在型順序を用いている．

(2) グレブナ基底 G の中から，$c_1,\ldots,c_{p+q+r+j}$ を含む要素を取り除く．このようにして得られた G の部分集合が，$c_k\neq 0$ に対応する特異点の軌跡 T_k を与える連立方程式となる． □

特異点が有限個数であればそれを計算する．特異点が無限個存在する場合には，その中に部分領域 A_{ij} に関する不等式制約 $t_{a_j(i,j+1)}>0,\ldots,t_{a_j(i,r)}>0$ を満足する点が存在するか否かを調べるため，次節で述べる方法に従って，さらに計算を進める．異なる k について同一の軌跡が重複して得られた場合には，それらのうちの1つについて計算を行う．

なお，場合によっては，得られた特異点の軌跡 T_k がもとの等式制約 (5) と完全に一致することがある．そのような場合には，それ以上特異点の計算を進めることができないので，計算が打ち切られる．特異点の軌跡 T_k がもとの方程式 (5) と一致するか否かは，それぞれのグレブナ基底を比較することにより検証される．

3.3 極値点が無限個存在する場合の再帰的計算

停留点もしくは特異点が無限個存在する場合には，それらの中に部分領域 A_{ij} に関する不等式制約 $t_{a_j(i,j+1)}>0,\ldots,t_{a_j(i,r)}>0$ を満足する点が存在するか否かを調べるために，さらに計算を進める必要がある．ここで提案する方法は，それら停留点の軌跡 S あるいは特異点の軌跡 T_k をあらためて等式制約とし，前節で述べた方法を再帰的に用いることによって有限個の実数解を特定するというものである．その際，停留点の軌跡 S を等式制約とする場合，目的関数を $u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ とは異なる別の関数 $u'(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ に変更する必要がある．そして，その新しい目的関数 $u'(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ には，$u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})=$一定 のときに恒等的には一定値とならない関数を選ばなくてはならない．なぜならば，停留点の軌跡 S は $u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})=$一定 となる軌跡であるので，$u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})=$一定 の下で恒等的に一定値となる関数を新しい目的関数として選んだ場合，求められる停留点の軌跡はやはり S となってしまうからである．再帰的計算の手順を以下に示す．

再帰的計算の手順

(1) 部分領域 A_{ij} に関する等式制約の集合を E とする．また，停留点の計算に用いられた目的関数の集合を U とし，初期条件として，U を空集合に設定する．

(2) E が有限個数の点であればそれを計算し，$t_{a_j(i,j+1)}>0,\ldots,t_{a_j(i,r)}>0$ を満足する点を選び出す．そして，計算を終了する．

(3) E の特異点の軌跡 T_k $(k=1,\ldots,m,\ m$ は E の要素数) を計算し，特異点軌跡の集合 T を作成する．

(4) 目的関数 $u'(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ を設定する．ただし，$u'(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ は任意の $u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})\in U$ について，$u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})=$一定 のときに恒等的には一定値とならないように設定する．そして，E のもとにおける $u'(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ の停留点の軌跡 S を計算する．

(5) すべての $T_k\in T$ および S について，以下の計算を行う．
- T_k について
  $E:=T_k$ として (2) へ進む．
- S について
  $E:=S$, $U:=U\cup\{u'(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})\}$ として (2) へ進む． □

実際のインプリメントでは，**図2** に示すように，目的関数 $u(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ を平行移動した関数を新しい目的関数 $u'(\boldsymbol{x},\boldsymbol{s},\boldsymbol{t})$ として用いるようにしている．

ここで，本再帰的計算の妥当性について考察しておく．部分領域 A_{ij} の開いた境界を B_{ij} とする．

$$B_{ij} = \{(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t}) | \prod_{m=j+1}^{r} t_{a_j(i,m)} = 0, \quad t_{a_j(i,j+1)} \geq 0, \ldots, t_{a_j(i,r)} \geq 0\}. \tag{10}$$

停留点の軌跡 S あるいは特異点の軌跡 T_k と，境界 B_{ij} との位置関係として，以下の3つが考えられる（**図3**）．

(1) S あるいは T_k が B_{ij} と交わらず，かつ，$t_{a_j(i,j+1)} > 0, \ldots, t_{a_j(i,r)} > 0$ なる領域に存在する（図3(1)）．

(2) S あるいは T_k が B_{ij} と交わらず，かつ，$t_{a_j(i,j+1)} > 0, \ldots, t_{a_j(i,r)} > 0$ なる領域に存在しない（図3(2)）．

(3) S あるいは T_k が B_{ij} と交わる（図3(3)）．

これら3つの場合のそれぞれについて，本再帰的計算によって得られる結果が妥当であることを示す．

- (1)の場合
  再帰的計算によって求められる $u'(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ の極値点は当然 $t_{a_j(i,j+1)} > 0, \ldots, t_{a_j(i,r)} > 0$ を満足するものであり，部分領域 A_{ij} における $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ の極値点が求められる．
- (2)の場合
  再帰的計算によって求められる $u'(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ の極値点は当然 $t_{a_j(i,j+1)} > 0, \ldots, t_{a_j(i,r)} > 0$ を満足しないものであり，部分領域 A_{ij} における $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ の極値点は存在しないとの結果が得られる．
- (3)の場合
  再帰的計算によって求められる $u'(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ の極値点が $t_{a_j(i,j+1)} > 0, \ldots, t_{a_j(i,r)} > 0$ を満足しないことがあり，部分領域 A_{ij} に $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ の極値点が存在するにもかかわらず，それが検出されない場合がある．しかしながら，このような場合には，$t_{a_j(i,m)} = 0$ $(j < m \leq r)$ との交点が，$t_{a_j(i,m)} = 0$ なる他の部分領域における停留点もしくは特異点として計算される．したがって，本論文では，再帰的計算によって得られた $u'(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ の極値点が $t_{a_j(i,j+1)} > 0, \ldots, t_{a_j(i,r)} > 0$ を満足しない点であった場合，A_{ij} に関してそれ以上の計算は行わない．

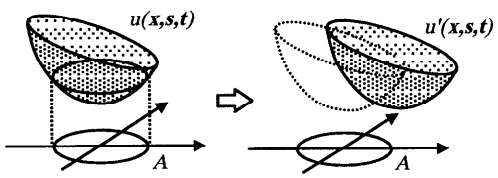


図2 目的関数の更新
Fig. 2 Renewal of the objective function.

3.4 計算手順

前節までに述べた手順をまとめたものを**図4**に示す．図中に現れる記号が示すものは以下のとおりである．

- i, j：部分領域を示す添字
- r：式(1)における等号を含む不等式制約 $h_k(\boldsymbol{x}) \geq 0$ の数
- A_{ij}：部分領域

まず i, j の初期値が与えられ，部分領域 A_{ij} として A_{10} が指定される．次に，部分領域 A_{ij} が有限個数の点であるか無限個数の点であるかが調べられる．有限個数であればその数値解が計算され，次の部分領域について計算を行う．無限個数であれば，3.2.1項および3.2.2項で述べた手順に従って，A_{ij} の特異点の軌跡ならびに目的関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ の停留点の軌跡が計算される．その後，3.3節で述べた手順に従い，解が有限個数となるまで同様の計算が再帰的に行われる．部分領域 A_{ij} に関する計算が終了すると，添字 i の値が $(i+1)$ に更新される $(i := i+1)$．更新された添字 i の値が ${}_rC_j$ 以下であるかどうかが調べられ，$i \leq {}_rC_j$ でない場合には添字 j の値が $(j+1)$ に更新されると同時に i には初期値1が設定される $(j := j+1,\ i := 1)$．その後，j の値が調べられる．j が r 以下でない場合には，未計算の部分領域は残っていないことになるので，すべての計算を終了する．

4. 例

本章では，4.1節において本論文で提案した手法のインプリメントについて述べ，その後，それを歯車軸

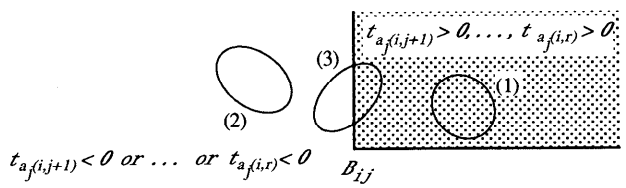


図3 停留点あるいは特異点軌跡と部分領域の境界
Fig. 3 Locus of stationary or singular points and boundary of sub-region.

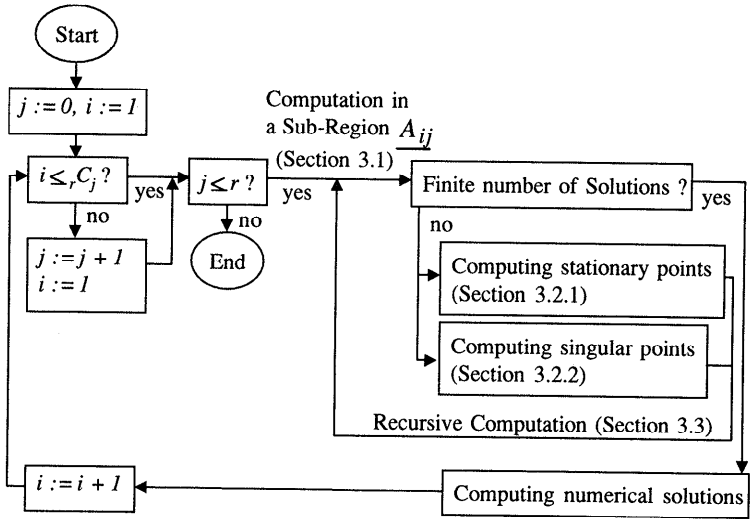


図4 計算手順
Fig. 4 Computing procedure.

のレイアウト問題に適用した例を 4.2 節に示す.

4.1 インプリメント

インプリメントの詳細は以下のとおりである.

(1) 言語および環境
数式処理ライブラリおよび言語には Risa/Asir[5),6)]を用いた.
また, OS は FreeBSD-2.2.1, CPU は MMX Pentium 166 MHz, メモリ容量は 80 MB である.

(2) 目的関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$
目的関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ は, 2 章の「目的関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$」で述べた条件を満足しさえすれば任意のものを選ぶことができる. その際, 係数および次数が小さい方が計算効率上有利であると考えられる. 本インプリメントでは, 目的関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ として以下のような二次関数を用い, 再帰的計算を行う場合にはこれを平行移動させる方法を採用した.

$$u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t}) = \sum_{k=1}^{p} x_k^2 + \sum_{k=1}^{q} s_k^2 + \sum_{k=1}^{r} t_k^2.$$

(3) 冗長な不等式制約の除去
与えられた問題に実数解が存在しない場合, すべての部分領域 R_{ij} について極値点の計算を行う必要がある. したがって, 冗長な不等式制約, すなわち他の不等式制約が満足されていれば自動的に満足されるような不等式制約は, 事前に取り除いておくことが計算効率上望ましい. 本インプリメントでは, 冗長性の検出が容易な 1 変数 1 次のものについては, 冗長な不等式制約を事前に取り除いている.

(4) 計算の終了条件
与えられた問題に実数解が存在する場合, 最初の数値解が求められた時点で計算を終了し, その解を返すようにした.

(5) 部分領域 A_{ij} の計算順序
与えられた問題に実数解が存在する場合, それを検出するのに要する時間はどの部分領域を先に計算するかによって大きく変わるものと考えられる. 解が有限個存在する部分領域を先に計算すれば, 極値点の計算を行わずに済ませられる分, 計算に要する時間が短くなるものと期待される. 解の個数が有限個か無限個かは, 単純に変数の数と等式制約の数から決まるものではないが, 経験上, 変数の数と等式制約の数が等しい場合は有限個, 変数の数が等式制約の数よりも多い場合は無限個, 変数の数が等式制約の数よりも少ない場合は零個である可能性が高いと考えられる. 本インプリメントでは, 変数の数と等式制約の数を比較し, これが等しい部分領域を先に計算し, 以下, 変数の数が等式制約の数よりも多い部分領域, 変数の数が等式制約の数よりも少ない部分領域, の順に計算するようにしてある.

(6) 解が有限個存在する連立方程式の実数解の計算
解が有限個存在する連立方程式が得られたとき, その実数解の計算には GSL 法[1)]を用いた (付録 A.2 参照). GSL 形式の計算には Risa/Asir

で提供されている組込関数を使い，また，GSL形式の中に現れる1変数方程式の実数解の計算にはスツルム関数列による2分法を用いた．

4.2 歯車箱レイアウト問題

例題として，**図5**に示す3軸歯車箱の，歯車軸の配置問題を考える．図中の各記号の意味は以下のとおりである．

- (x_j, y_j) $(j = 1, 2, 3)$：第 j 軸の位置座標
- d_{ji}，d_{jo} $(j = 1, 2, 3)$：第 j 軸歯車の入力側直径および出力側直径
- w：歯車箱の横幅
- h：歯車箱の高さ

第1軸に入力された回転力は，第2軸を経由して，出力軸である第3軸へ伝達される．この問題は，歯車径と箱の大きさが与えられたときに歯車軸の位置を求めるものである．歯車箱に十分な大きさがあれば，図5に示す歯車どうしのかみ合い関係を保ったままですべての歯車を歯車箱の中に配置することが可能であり，解は無限個存在する．一方，歯車箱に十分な大きさがなければ，図5に示す歯車どうしのかみ合い関係を保ったまますべての歯車を歯車箱の中に配置することはできず，解は存在しない．制約条件は次式(11)で与えられる．

$$
\begin{aligned}
&(x_2 - x_1)^2 + (y_2 - y_1)^2 = ((d_{1o} + d_{2i})/2)^2,\\
&(x_3 - x_2)^2 + (y_3 - y_2)^2 = ((d_{2o} + d_{3i})/2)^2,\\
&(x_1 - x_3)^2 + (y_1 - y_3)^2 \geq ((d_{1o} + d_{3i})/2)^2,\\
&d_{1o}/2 \leq x_1 \leq w - d_{1o}/2,\\
&d_{1o}/2 \leq y_1 \leq h - d_{1o}/2,\\
&d_{2i}/2 \leq x_2 \leq w - d_{2i}/2,\\
&d_{2o}/2 \leq x_2 \leq w - d_{2o}/2,\\
&d_{2i}/2 \leq y_2 \leq h - d_{2i}/2,\\
&d_{2o}/2 \leq y_2 \leq h - d_{2o}/2,\\
&d_{3i}/2 \leq x_3 \leq w - d_{3i}/2,\\
&d_{3i}/2 \leq y_3 \leq h - d_{3i}/2.
\end{aligned}
\tag{11}
$$

式(11)の第1式は第1軸歯車と第2軸歯車のかみ合い条件，第2式は第2軸歯車と第3軸歯車のかみ合い条件である．第3式は第1軸歯車と第3軸歯車とが互いに干渉しないための条件を示し，残りの式は各歯車が箱の中に収まるための条件を示している．

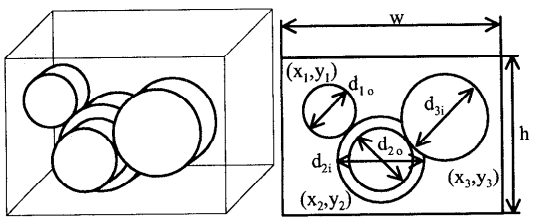


図5 3軸歯車箱
Fig. 5 Three-axes gearbox.

ここで，歯車径を次式(12)のように与え，箱の大きさ w，h を変えて計算を行い，本手法により過少代数制約問題の数値解を得られることを確認した．

$$
\begin{aligned}
&d_{1o} = 42\,[\mathrm{mm}],\ d_{2i} = 86\,[\mathrm{mm}],\\
&d_{2o} = 70\,[\mathrm{mm}],\ d_{3i} = 110\,[\mathrm{mm}].
\end{aligned}
\tag{12}
$$

(1) $w = 180\,[\mathrm{mm}]$，$h = 180\,[\mathrm{mm}]$ の場合
実数解が存在し，その例として以下の数値解が計算された．計算時間は15.9秒で，そのうちCPU時間が11.9秒，GC時間が4秒であった．

$$(x_1, y_1, x_2, y_2, x_3, y_3) = (97.4, 54.2, 43, 87.9, 125, 125)$$

(2) $w = 160\,[\mathrm{mm}]$，$h = 160\,[\mathrm{mm}]$ の場合
数値解は得られず，したがって，実数解は存在しないと示される．その計算時間は2940秒で，そのうちCPU時間が2341秒，GC時間が599秒であった．

従来，このようなレイアウト問題は試行錯誤によって解くしかなく，特に解が得られない場合，それが解がそもそも存在しないためなのか，存在している解を発見できないためなのかを見極めることがきわめて困難であった．本手法によりそれらの困難が解消され，この例題のように箱の大きさがわずかに小さくなるだけで解が存在しなくなるような場合でも，過少代数制約を適切に評価して実数解を得ることが可能となった．

5. おわりに

本論文の例にもあるように，設計問題やレイアウト問題は多くの場合過少制約問題であり，解は一意には決まらない．その解決を支援するためには，以下の機能を提供することが有効であると考えられる．

(1) 解が存在する場合には，解の例を示す．
(2) 解が存在しない場合には，解が存在しないことを示す．

これらの機能を提供するため，本論文では，過少代数制約問題の数値解を数式処理的手法を用いて求める方法を提案した．そして，それを歯車箱の配置問題に適用した例を示した．本論文で提案した手法を用いることには以下の利点がある．

- 問題解決にあたる作業者は，それらの例示された解を参照・修正し，あるいは，新しく制約条件を加えることによってより好ましい解を得ることができる．
- 問題解決にあたる作業者は，解が存在しないこと

が分かれば，その時点で作業を打ち切ることができ，無駄な労力を費さずに済ませられる．

なお，本論文で示した例では，最初の数値解が得られた時点で計算を打ち切るようにしたが，計算される解の個数には任意の値を設定することが可能である．

今後の課題として以下があげられる．

(1) 特異点解析手法の開発

本手法では，3.2.2 項ですでに述べたように，特異点の軌跡が元の軌跡と完全に一致してしまった場合，それ以上の計算は行われていない．したがって，そのような計算不能特異点の中に存在している実数解を見落とす可能性は否定できず，その意味において，本手法が不完全なものであることは認めざるをえない．本手法を完全なものとするためには，さらに別の特異点解析手法を開発する必要がある．

(2) 計算効率の向上

本手法をより大規模な問題，すなわち，変数や制約条件がより多く存在する問題に適用してゆく場合には，計算効率の向上が不可欠である．計算効率向上のために検討すべき項目として以下が考えられる．

- グレブナ基底計算に関連する部分
  本手法では，グレブナ基底およびそれに関連する計算が多く行われる．したがって，本手法の効率を向上させるためには，それらの計算効率向上が必要である．グレブナ基底計算では，多項式の係数が巨大となるために計算効率が低くなる場合が多いので，近似的手法[9]の導入などで，計算の高速化をはかる必要がある．
- 目的関数の選択
  本手法では，目的関数 $u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ の選び方が大きく影響してくるものと考えられる．現在のところ，4.1 節で述べたように，$u(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{t})$ には各変数の自乗の和を用いているが，他の関数についても検討する必要がある．
- 不等式の表現
  等号を含まない不等式 $e(\boldsymbol{x}) > 0$ は，与えられた制約条件を閉領域として扱うために，$e(\boldsymbol{x}) \neq 0 \wedge e(\boldsymbol{x}) \geq 0$ と表現されている．しかしながら，この表現方法では計算に使われる式および変数の数が増え，計算量が増すことになるので，別の表現方法を考える必要がある．

謝辞 本研究にあたり，適切な助言をいただいた機械技術研究所極限技術部長の小鍛治繁博士に深く感謝します．また，GSL 法および Risa/Asir でのインプリメントに関し助言をいただいた富士通研究所の野呂正行氏にこの場を借りて感謝申し上げます．

参考文献

1) Alonso, M.-E., Becker, E., Roy, M.-F. and Wörmann, T.: Zeros, multiplicities, and idempotents for zero-dimensional systems, *Progress in Mathematics*, Vol.143, pp.1–15 (1996).
2) Buchberger, B.: Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory, *Multidimensional Systems Theory*, chapter 6, Bose, N.K. (Ed.), pp.184–232, D. Reidel Publishing Company (1985).
3) Hong, H. and Liska, R.: Special Issue on Application of Quantifier Elimination: Forward of the Guest Editors, *Journal of Symbolic Computation*, Vol.24, No.2, p.123 (1997).
4) Lazard, D.: Gröbner-bases, Gaussian elimination and resolution of systems of algebraic equations, *Proc. EUROCAL '83*, Lecture notes in Computer Science, Vol.162, pp.146–156, Springer-Verlag (1983).
5) Noro, M. and Takeshima, T.: Risa/Asir – A Computer Algebra System, *Proc. ISSAC '91*, pp.387–396, ACM (1992).
6) Noro, M. and Takeshima, T.: High-Quality Computing of Polynomial Problems by Risa/Asir, *Fujitsu Science and Technology Journal*, Vol.32, No.2, pp.256–270 (1996).
7) 坂和正敏：非線形システムの最適化〈一目的から多目的へ〉，p.27，森北出版 (1986).
8) 佐々木健昭，今井 浩，浅野孝夫，杉原厚吉：岩波講座 応用数学：計算代数と計算幾何，p.6，岩波書店 (1993).
9) Sasaki, T.: A study of approximate polynomials. I: Representation and arithmetic, *Japan Journal of Industrial and Applied Mathematics*, Vol.12, No.1, pp.137–161 (1995).
10) 上野健爾：代数幾何入門，p.133，岩波書店 (1995).
11) Weispfenning, V.: Quantifier Elimination for Real Algebra – The Quadratic Case and Beyond, *Applicable Algebra in Engineering, Communication and Computing*, Vol.8, pp.85–101 (1997).
12) Weispfenning, V.: Simulation and Optimization by Quantifier Elimination, *Journal of Symbolic Computation*, Vol.24, No.2, pp.189–208 (1997).

付　　録

A.1 項順序[8)]

n 個の非負整数からなる組 $A=(\alpha_1,\ldots,\alpha_n)$ と $B=(\beta_1,\ldots,\beta_n)$ に対し，A と B の順序 $\succ$ を次のように定める．なお，$A\succ B$ は，A が B よりも高順位であることを示す．

$$A\succ B \Leftrightarrow \{\alpha_1>\beta_1\}\vee \exists(k,1<k\leq n)\{\alpha_i=\beta_i\ (i=1,\ldots,k-1)\wedge \alpha_k>\beta_k\}.$$

この順序を用いて，単項式に順序を導入できる．この単項式間の順序を項順序という．単項式 $T=cz_1^{e_1}\ldots z_n^{e_n}$ (c は係数) を指数の組 $(e_1,\ldots,e_n)$ に対する順序 $\succ$ で順序付けるとき辞書式順序といい，$(\sum_{i=1}^{n}e_i,\sum_{i=1}^{n-1}e_i,\ldots,e_1)$ で順序付けるとき全次数逆辞書式順序という．$z_1,\ldots,z_n\succ y_1,\ldots,y_m$ に関する辞書式順序と，$z_1\succ\ldots\succ z_n$ ならびに $y_1\succ\ldots\succ y_m$ に関する全次数逆辞書式順序を組み合わせた混在型順序とは，単項式 $T=cz_1^{e_1}\ldots z_n^{e_n}y_1^{d_1}\ldots y_m^{d_m}$ を $(\sum_{i=1}^{n}e_i,\sum_{i=1}^{n-1}e_i,\ldots,e_1,\sum_{j=1}^{m}d_j,\sum_{j=1}^{m-1}d_j,\ldots,d_1)$ で順序付けるものである．

このような項順序の選び方によってグレブナ基底の計算時間に大きな差が出ることはよく知られている[4)]．一般に，辞書式順序を用いた場合には，全次数逆辞書式順序を用いた場合に比べてはるかに多くの計算時間を要する．

A.2 GSL 法[1)]

実数係数の多項式からなる連立方程式 (13) が与えられており，その解の個数は有限個であるものとする．

$$\phi_1(z_1,\ldots,z_n)=\ldots=\phi_m(z_1,\ldots,z_n)=0. \tag{13}$$

連立方程式 (13) の解の個数を l とし，それらの解を $\boldsymbol{a^i}$ $(i=1,\ldots,l)$ と表すことにする．すなわち，次式 (14) が成立するものとする．

$$\phi_1(\boldsymbol{a^i})=\ldots=\phi_m(\boldsymbol{a^i})=0\ (i=1,\ldots,l). \tag{14}$$

ここで，空間上に新しい座標軸 y を導入し，$(z_1,\ldots,z_n)$ から y 軸への写像 ψ を考える（式 (15)）．

$$\psi:(z_1,\ldots,z_n)\to y,\quad \psi(z_1,\ldots,z_n)=\alpha_1z_1+\ldots+\alpha_nz_n \tag{15}$$
$$(\alpha_1,\ldots,\alpha_n \text{ は実数}).$$

もしも，連立方程式 (13) の解 $\boldsymbol{a^i}$ と，写像 ψ による $\boldsymbol{a^i}$ の像 $\psi(\boldsymbol{a^i})$ との対応関係が 1 対 1 であるならば，各変数 z_j を y の関数として表現した次式 (16) の形の連立方程式を導くことが可能であり，これを用いてもとの連立方程式 (13) の解を求めることができる．

$$\begin{aligned}\varphi(y)&=0,\\ z_1&=\varphi_1(y),\\ &\vdots\\ z_n&=\varphi_n(y).\end{aligned} \tag{16}$$

ただし，$\varphi(y)$ は実数係数の多項式，$\varphi_j(y)$ $(j=1,\ldots,n)$ は実数係数の有理式である．辞書式順序のグレブナ基底は式 (16) の形を持つ典型的な例であり，このとき $\varphi_j(y)$ は実数係数の多項式となる．

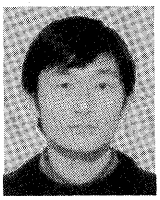
GSL 形式とは，式 (16) において有理式 $\varphi_j(y)$ が次式 (17) の形をとるものである．

$$\varphi_j(y)=v_j(y)/\varphi'(y)\ (j=1,\ldots,n). \tag{17}$$

ただし，$v_j(y)$ は実数係数の多項式，$\varphi'(y)$ は $\varphi(y)$ の y による 1 階微分を示す．GSL 形式は，一般に辞書式順序のグレブナ基底に比べて係数が小さく，計算効率の面で優れているとされる．本論文で採用している GSL 法とは，GSL 形式を計算し，これを用いて実数解を計算するものである．

Risa/Asir では，y 軸として $z_1,\ldots,z_n$ 軸のいずれかをユーザが指定し，その指定に基づいて GSL 形式の計算を行う組込関数が提供されている．指定された軸 z_j $(j=1,\ldots,n)$ が適切でなかった場合，すなわち，連立方程式の解 $\boldsymbol{a^i}$ とその z_j 軸座標値とが 1 対 1 に対応しなかった場合，Risa/Asir は GSL 形式の代わりに辞書式順序のグレブナ基底を計算する．

(平成 10 年 5 月 13 日受付)
(平成 11 年 2 月 8 日採録)



沢田　浩之（正会員）

1964 年生．87 年東京大学工学部航空学科卒業．89 年，東京大学大学院工学系研究科航空学専攻修士課程修了．同年工業技術院機械技術研究所勤務．90 年より 92 年まで（財）新世代コンピュータ技術開発機構へ休職出向．同年，機械技術研究所に復職．数式処理およびその機械設計支援技術への応用に取り組む．日本機械学会，日本数式処理学会各会員．