数理解析研究所講究録
第1907巻 2014年 20-31



# パラメータ入りの連立線形方程式の誤差低減法

# A Method of Reducing Numerical Errors in Parametric Linear Systems Solving

佐々木 建昭 (Tateaki Sasaki) *
筑波大学 名誉教授

山口 哲 (Tetsu Yamaguchi) †
Maplesoft, Canada

**Abstract**

先端産業分野では製品開発において、製品に対する数理モデルを物理法則と拘束条件の集合で表現し、モデルの挙動を数値シミュレーションして、そのモデルを規定する変量の最適値を決めていく設計方法が盛んになりつつある。しかし、複合的な製品ではモデルが大きくなり、モデル変量を全てパラメータとして計算するのは非現実的である。そこで、物理定数や適値を推定できるパラメータは浮動小数で与えて計算量を減らすことが考えられるが、そうすると今度は計算中の誤差をいかに低減化するかが大きな問題となる。本稿では、数理モデルに基づく設計で現れるパラメータ入りの疎線形方程式系を念頭に、筆者らが最近提案し国際会議で発表した誤差低減法を簡単に記述し、算法を自己批判するとともに、今後の課題について述べる。

## 1 まえがき、特に“モデルに基づく開発”について

モデルに基づく開発 (Model-based Development: MBD) は一昔前、LSIの回路設計において盛んだったが、現在、他の多くの先端産業においても盛んになりつつある。乗用車や航空機などの機械の動きは物理法則と形状等に関する制約の下に設計されるが、後者の制約を代数方程式系で表すなら、その機械は微分代数方程式系 (DAE系) として数学的にモデル化される；ここで微分は時間$t$に関するものである。設計とは、この系に含まれるパラメータの最適値の組を決定することである。実際に最適値を決めるには、パラメータに適当な数値を代入して方程式系の挙動を数値シミュレーションし、その結果を解析してパラメータの値を修正し、これを繰り返して最適値に近づける。

*sasaki@math.tsukuba.ac
†tetsuy@maplesoft.ca

しかし、この方法をそのまま実行しても、計算は多くの場合うまく行かない。そのことを文献[2]に基づいて説明する。第一に、DAE系は数値計算にとって極端な悪条件系で、まともには解けない。そこで考案されたのが、代数方程式を必要な回数だけ微分してDAE系を常微分方程式系(ODE系)に変換する方法である。しかし、得られたODE系はそのままではシミュレーションできない。A, B, C, ... 順に与えられた(微分)方程式は、Aを数値的に決めるにはBの値が必要で、Bを数値的に決めるにはCの値が必要で、... というように並んでいることがほとんどで、これを前から順に計算できる順序に並べ替える必要がある；これを因果化(causalization)という。

DAE系からODE系への変換では、微分多項式（変量$x$や$y$、それらの時間微分$\dot{x}$や$\dot{y}$などを変数として含む多項式）の消去、いわゆる微分消去が行われる。これはグレブナー基底計算における項消去のようなもので（ただし、消去後の式が$x$や$y$の多項式になるように$x$や$y$の単項式を掛ける）、浮動小数係数で計算すれば当然大きな誤差が生じ得る。その際、どんなメカニズムで大きな誤差(桁落ち誤差)が生じ、それをどう回避するかについては筆者らの国際会議論文[14]を参照されたい。

因果化では困ることは何も起きないように思われるが、実は大きな問題が頻繁に生じることが1960年代から知られている[2]。与えられた方程式系の部分集合$\{E_{i_1}, \ldots, E_{i_k}\}$には、その数式を見ただけでは因果的に並べ替えることができず、それらに含まれる変量$x_1, \ldots, x_m$について解いて初めて並べかえることができる、そんなものがあるのである。簡単のため(多くの場合が該当する)、$x_1, \ldots, x_m$がこれらの方程式の中に1次で現れるとする（たとえば$x^2$が現れれば$x^2$を一つの変量とみなす）。方程式の係数はパラメータであり、因果化は数値シミュレーションに先立って実行されるので、我々はパラメータ係数の線形方程式系を解くことになる。いくつかのパラメータや物理・工学定数を浮動小数で近似すれば、計算過程で生じる誤差対策が不可欠である。

線形方程式系に関しては、係数が数値の場合は研究され尽くされたと言っても過言ではないほどだが、パラメータ混じりの係数の場合にはまだ研究の余地がある。MBDでは与式はいずれも非常に疎だが、パラメータ係数連立線形疎方程式の解法は次論文に譲り、本稿では疎という性質は考慮しない。仮定するのは係数の多くは浮動小数でその中にパラメータ係数が混じっている、ということだけである。本稿では、線形方程式系の係数行列をGauss-Jordan消去していくが、各消去後の行列の要素を元の行列要素の行列式で表す多くの公式をBareiss[1]に従って導出する。これら公式は読者に有用だろうと思うので、導出も含めてすべて掲載する。

第2章では、我々の方法の概要を説明する。それは数値消去と記号行列式の計算を混用する“数値数式混合解法”である。第3章では、数値消去中の要素を行列式で表す公式を前進消去と後退消去各々に対して導出する。国際会議論文[14]では後退消去に関する一部の公式導出が解り難いので、解り易く記述する。第4章では、数値消去後に残る記号行列の消去における行列式公式(Cramer公式)を導出し、数値数式混合解法のための解公式を導出する。そして、Cramer公式に対する分割征服Laplace展開法を記述する。第5章では、本稿に書き切れなかったことを注釈の形で記述する。第6章では、本研究のまとめと批判および今後の課題を述べる。

## 2 本稿で呈示する方法の概要

与えられた線形方程式系を次式とする；$\mathbb{F}$ は固定精度浮動小数の集合を表す。

$$C\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{x} = (x_1, \ldots, x_m)^{\mathrm{t}}, \quad C \in \mathbb{F}[\boldsymbol{p}]^{m\times m}, \tag{2.1}$$

$$C = \begin{pmatrix} c_{1,1} & \cdots & c_{1,m} \\ \vdots & \ddots & \vdots \\ c_{m,1} & \cdots & c_{m,m} \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}. \tag{2.2}$$

ここで、$\boldsymbol{p}$ はパラメータの集合である。以下では定数項ベクトル $\boldsymbol{b}$ は行列 $C$ の第 $(m+1)$-列として扱い、$(c_{m,m+1}, \ldots, c_{m,m+1})^{\mathrm{t}}$ と表す。

係数行列 $C$ は既に次のように行と列が並べ替えられているとする。

$$\text{仮定 } C: \begin{cases} \text{左側の } n \text{ 列は数値列 } (n < m), \\ \text{上側の } n' \text{行は数値行 } (n' \le n), \\ \forall k \le n \text{ に対し、主座行列式} \neq 0. \end{cases} \tag{2.3}$$

(注) $n < n'$ であっても良いが、その場合は下記とは異なり、行消去を行う。

我々の方法（というほど斬新ではなく平々凡々とした方法であるが）は、まず数値消去を可能な限り行う。ここで数値消去とは行 (または列) に数値を掛けることで列消去 (または行消去) を行うことである。上記の仮定 C に即して言えば、左 $n$ 列 (あるいは上 $n'$ 行) の消去は数値を掛けることで行えて、消去の結果、パラメター要素の次数は上昇しない（数値要素がパラメータ要素になることはある)。その方法として我々は、よく知られた pivoting Gauss-Jordan 法を採用する：Gauss-Jordan 法とは、Gauss 法が行列を上三角型に変形し次いで後方代入で対角化するのに対して、各列消去の直後に行消去を行い対角化するものである。なお、数値行列の消去においては QR 分解が誤差の集積を抑える方法として有名だが、QR 分解では行 (または列) に掛ける "乗数" を消去にからむ行 (または列) の全要素から計算するので、パラメータ入り行列ではうまく働かない。

上記の数値消去後、$(m-n)\times(m-n+1)$ 記号行列が得られる。この記号行列も Gauss 消去することが可能だが（ただし、消去結果が有理式にならないよう、たとえば行 $(e_{21}, e_{22}, \ldots)$ を行 $(e_{11}, e_{12}, \ldots)$ で消去するには $e_{11}\times(e_{21}, e_{22}, \ldots) - e_{21}\times(e_{11}, e_{12}, \ldots)$ と行う)、計算量が大きすぎるのみならず、特別な技法を使わなければ大きな桁落ち誤差が発生することが知られている [4, 12, 13]。浮動小数係数の記号行列式の計算では、蜜行列に対してさえ、計算量および桁落ち誤差の双方から小行列式展開法がベストであることが幾つかの実験例で知られている [4, 13]。そこで我々は、記号行列の線形方程式系の解を Cramer 公式を Laplace 展開することで計算する。ただし、Cramer 公式に現れる行列式は、行列を左側と右側にほぼ等分すると同じ小行列が多く現れるので、同じ小行列式を何度も計算しないように分割征服法で計算する。

この解法では、前半の数値消去と後半の行列式展開法を結ぶ公式が必要であり、次章でその公式を導出する。この公式を見れば、数値消去のどの段階でどのようなメカニズムにより桁落ち誤差が入り込むかが一目瞭然であり、低減策も簡単に思いつく。

# 3 Gauss-Jordan法による数値消去の行列式理論

本章の行列式理論はBareiss[1]によるよく知られたもので、筆者らのオリジナルなものではない。誤差発生メカニズムと後退消去における行列式表現は筆者らのものである。

$C_{0,0}^{(-1)}=1$, $C_{i,j}^{(0)}=c_{i,j}$ $(\forall i,j)$ とし、$C_{i,j}^{(k)}$ $(k\geq 1)$ を次の行列式とする：

$$C_{i,j}^{(k)} \stackrel{\text{def}}{=} \begin{vmatrix} c_{1,1} & \cdots & c_{1,k} & c_{1,j} \\ \vdots & \ddots & \vdots & \vdots \\ c_{k,1} & \cdots & c_{k,k} & c_{k,j} \\ c_{i,1} & \cdots & c_{i,k} & c_{i,j} \end{vmatrix} \quad (1\leq k<\forall i,j). \tag{3.1}$$

このとき、次の恒等式が成立する：本章ではこの恒等式だけを用いて行列式理論を作る。

$$C_{i,j}^{(k)} = \begin{vmatrix} C_{k,k}^{(k-1)} & C_{k,j}^{(k-1)} \\ C_{i,k}^{(k-1)} & C_{i,j}^{(k-1)} \end{vmatrix} / C_{k-1,k-1}^{(k-2)}. \tag{3.2}$$

$k$段目の前進消去後の $m\times(m+1)$ 行列の第 $(i,j)$ 要素が (3.1) で表せることはよく知られている。以下では、Row$(i)$ とは消去中の $m\times(m+1)$ 行列の第 $i$ 行を表すものとする。

## 3.1 前進消去における項キャンセルと誤差回避策

前章で述べたように、実際の算法では、前進消去と後退消去を交互に繰り返しながら、1要素毎に対角化していくが、説明の便宜上、本節では前進消去だけを記述する。

最初の消去は次のように行う $(2\leq i\leq m)$。

$$\begin{aligned} \text{Row}(i) \Rightarrow\ & c_{1,1}\times\text{Row}(i)-c_{i,1}\times\text{Row}(1) \\ =\ & (\,0\,,\,C_{i,2}^{(1)}\,,\,\cdots\,,\,C_{i,m+1}^{(1)}\,),\ \text{where} \\ C_{i,j}^{(1)} =\ & \begin{vmatrix} c_{1,1} & c_{1,j} \\ c_{i,1} & c_{i,j} \end{vmatrix} \quad (2\leq j\leq m+1). \end{aligned} \tag{3.3}$$

第2段目の消去は次のように行う $(3\leq i\leq m)$。

$$\begin{aligned} \text{Row}(i) \Rightarrow\ & \{C_{2,2}^{(1)}\times\text{Row}(i)-C_{i,2}^{(1)}\times\text{Row}(2)\}/c_{1,1} \\ =\ & (\,0\,,\,0\,,\,C_{i,3}^{(2)}\,,\,\cdots\,,\,C_{i,m+1}^{(2)}\,),\ \text{where} \\ C_{i,j}^{(2)} =\ & \begin{vmatrix} c_{1,1} & c_{1,2} & c_{1,j} \\ c_{2,1} & c_{2,2} & c_{2,j} \\ c_{i,1} & c_{i,2} & c_{i,j} \end{vmatrix} \quad (3\leq j\leq m+1). \end{aligned} \tag{3.4}$$

もしも $C_{i,j}^{(2)}$ をピボッティングなしで上式の先頭行のように計算するなら、大きな誤差が頻繁に生じるであろう。その理由を $c_{i,j}\in\mathbb{F}$ の場合について説明する。先頭行の分子は $(c_{1,1}c_{2,2}-c_{2,1}c_{1,2})\times(c_{1,1}c_{i,j}-c_{i,1}c_{1,j})-(c_{1,1}c_{i,2}-c_{i,1}c_{1,2})\times(c_{1,1}c_{2,j}-c_{2,1}c_{1,j})$ と計算されるが、積 $\tilde{c}\stackrel{\text{def}}{=}c_{2,1}c_{1,2}c_{i,1}c_{1,j}$ は正確にキャンセルする。しかしながら、$\tilde{c}$ はキャンセルする

前に積 $c_{1,1}c_{2,2}c_{1,1}c_{i,j}$ 等に加えられている。したがって、もしも $\tilde{c}$ が他の積よりも遥かに大きければ、$\tilde{c}$ の誤差がキャンセルしない積に入り込むので、結果式の相対誤差が大きく増加するのである。我々が扱う行列は $c_{i,j} \in \mathbb{F}[\boldsymbol{p}]$ だが、この場合でも上記メカニズムで誤差が発生する。このメカニズムは組織的なので組織的項キャンセルと呼ぶ。

数値消去で誤差増大を避ける最も有効かつ簡単な方策はピボッティングだが、ピボットの選択は賢明に行う必要がある。通常、ピボットは大きい数値要素を選ぶ。しかし、Row($i$) の記号要素の係数が他行の記号要素に比べて大きいとき、$c_{i,i}$ をピボットに選んで Row($i$) で他行を数値消去すると消去後の他行の各記号要素は Row($i$) に似たものとなり、たとえ行列式を小行列式展開法で計算しても大きな桁落ち誤差が発生する。ピボットは記号要素の大きさも考慮して選ぶ必要がある。

さて、$k$ 段目の消去 ($k \geq 3$) を考える。$k = 1 \Rightarrow 2 \Rightarrow \cdots$ と消去を進め、$k$ 段目の消去として (3.2) により Row($k+1$), ..., Row($m$) を Row($k$) で消去するなら、消去後の行列の $(i,j)$ 要素は式 (3.1) の行列式 $C^{(k)}_{i,j}$ となる。問題は如何に組織的項キャンセルを避けるかである。本節最後で述べるように、$k \geq 4$ のとき全ての組織的項キャンセルを回避するのは非常に厄介で実際的でない。本章では不完全だが簡単な方法を考える。

上に述べたように、$k = 2$ の場合には組織的項キャンセルを避けることができた。同じことをすれば大きな項キャンセルを避けられるであろう。即ち、$C^{(k-2)}_{i',j'}$s ($\forall i', j' \geq k-1$) から $C^{(k)}_{i,j}$ を計算するのである。$\widetilde{C}^{(k-1)}_{i',j'}$ と $\widetilde{C}^{(k)}_{i,j}$ を下記のように計算したとしよう。

$$\begin{aligned} \widetilde{C}^{(k-1)}_{i',j'} &:= \begin{vmatrix} C^{(k-2)}_{k-1,k-1} & C^{(k-2)}_{k-1,j'} \\ C^{(k-2)}_{i',k-1} & C^{(k-2)}_{i',j'} \end{vmatrix}, \\ \widetilde{C}^{(k)}_{i,j} &:= \begin{vmatrix} \widetilde{C}^{(k-1)}_{k,k} & \widetilde{C}^{(k-1)}_{k,j} \\ \widetilde{C}^{(k-1)}_{i,k} & \widetilde{C}^{(k-1)}_{i,j} \end{vmatrix} / C^{(k-2)}_{k-1,k-1}. \end{aligned} \tag{3.5}$$

このとき、$\widetilde{C}^{(k)}_{i,j}$ の計算では大きな項キャンセルが起きる。しかしながら、$\widetilde{C}^{(k)}_{i,j}$ の行列式は $C^{(k-2)}_{i',j'}$ ($i', j' \geq k-1$) を要素とする 3×3 行列式に変換できるので、その行列式を計算するのである。$\widetilde{C}^{(k-1)}_{i',j'}$ と $\widetilde{C}^{(k)}_{i,j}$ はそれぞれ $[C^{(k-3)}_{k-2,k-2}]C^{(k-1)}_{i',j'}$ と $[C^{(k-3)}_{k-2,k-2}]^2 C^{(k)}_{i,j}$ に等しいので、次の公式が得られる。

$$C^{(k)}_{i,j} = \begin{vmatrix} C^{(k-2)}_{k-1,k-1} & C^{(k-2)}_{k-1,k} & C^{(k-2)}_{k-1,j} \\ C^{(k-2)}_{k,k-1} & C^{(k-2)}_{k,k} & C^{(k-2)}_{k,j} \\ C^{(k-2)}_{i,k-1} & C^{(k-2)}_{i,k} & C^{(k-2)}_{i,j} \end{vmatrix} / [C^{(k-3)}_{k-2,k-2}]^2 \qquad (3 \leq k < \forall i, j). \tag{3.6}$$

これは Bareiss の 2-step 公式に他ならない。この算式は、$[C^{(k-3)}_{k-2,k-2}]^2$ に比例しない項が正確にキャンセルすることを示している。したがって、$C^{(k-3)}_{k-2,k-2}$ が相対的に小さいときは桁落ち誤差が発生する。これを回避するには 4×4 行列式を計算すればよいが、そこではまた別の項キャンセルが発生する。組織的項キャンセルを全て回避するのが厄介なことが分るであろう。

## 3.2 後退消去における項キャンセルと誤差回避策

Gauss-Jordan 法は前進消去を 1 段行った後に先頭要素を対角化するが、対角化演算を後退消去 と呼ぶ。そこでも組織的項キャンセルが起きることが分るだろう。

2 段目の消去に先立ち、第 1 段消去後の行列の第 1 行を次のように書き換える。

$$(C^{(1)}_{1,1}, C^{(1)}_{1,2}, C^{(1)}_{1,3}, \cdots) \stackrel{\text{def}}{=} (c_{1,1}, c_{1,2}, c_{1,3}, \cdots) = \left(c_{1,1}, \begin{vmatrix} c_{1,1} & c_{1,2} \\ -1 & \end{vmatrix}, \begin{vmatrix} c_{1,1} & c_{1,3} \\ -1 & \end{vmatrix}, \cdots\right). \quad (3.7)$$

(3.1) の最下行を $(-1, 0, \cdots)$ で置き換えた行列に公式 (3.2) を適用すると、(3.4) で示したように、Row(1) を Row(2) により次のように消去できる。

$$\left(c_{1,1}\begin{vmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{vmatrix}, 0, \begin{vmatrix} C^{(1)}_{2,2} & C^{(1)}_{2,3} \\ C^{(1)}_{1,2} & C^{(1)}_{1,3} \end{vmatrix}, \cdots\right)/c_{1,1} = \left(\begin{vmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{vmatrix}, 0, \begin{vmatrix} c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,1} & c_{2,2} & c_{2,3} \\ -1 & & \end{vmatrix}, \cdots\right). \quad (3.8)$$

上記の左辺のように計算すれば組織的項キャンセルが起きるが、右辺のように計算すれば起きない。(3.8) の右辺を $(C^{(2)}_{1,1}, 0, C^{(2)}_{1,3}, C^{(2)}_{1,4}, \cdots)$ と定義する。帰納法により、$k$ 段目の後退消去後の Row(1) を $(C^{(k)}_{1,1}, \cdots, C^{(k)}_{1,k+1}, \cdots)$ とすれば、$C^{(k)}_{1,j}$ は次式となる。

$$C^{(k)}_{1,j} = \begin{vmatrix} c_{1,1} & \cdots & c_{1,k} & c_{1,j} \\ \vdots & \ddots & \vdots & \vdots \\ c_{k,1} & \cdots & c_{k,k} & c_{k,j} \\ -1 & & & \end{vmatrix} \quad (\forall j > k \geq 2). \quad (3.9)$$

次に第 3 段目の後退消去を考える。消去に先立ち、Row(2) を次のように書き換える。

$$(\,0\,, C^{(2)}_{2,2}, C^{(2)}_{2,3}, C^{(2)}_{2,4}, \cdots) \stackrel{\text{def}}{=} \left(\,0\,, C^{(1)}_{2,2}, \begin{vmatrix} c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,1} & c_{2,2} & c_{2,3} \\ & -1 & \end{vmatrix}, \begin{vmatrix} c_{1,1} & c_{1,2} & c_{1,4} \\ c_{2,1} & c_{2,2} & c_{2,4} \\ & -1 & \end{vmatrix}, \cdots\right). \quad (3.10)$$

公式 (3.2) を用いると、これを (3.4) の Row(3) で消去した Row(2) の第 $j$ 要素は次のように表せる。

$$C^{(3)}_{2,j} = \begin{vmatrix} c_{1,1} & c_{1,2} & c_{1,3} & c_{1,j} \\ c_{2,1} & c_{2,2} & c_{2,3} & c_{2,j} \\ c_{3,1} & c_{3,2} & c_{3,3} & c_{3,j} \\ & -1 & & \end{vmatrix} \quad (\forall j > 3). \quad (3.11)$$

この行列式表現は、(3.9) の $C^{(k)}_{1,j}$ と全く同様に、$C^{(k)}_{2,j}$ $(k \geq 4)$ と $C^{(k)}_{i,j}$ $(i < k < j)$ に容易に一般化できる。なお、実際の計算では $k > 3$ に対する行列式公式は使わない。

さて、後退消去における組織的項キャンセルの回避策を考える。我々の方策は、もしも (3.2) のような行列式に遭遇したなら、(3.6) のような 3×3 行列式に変換することである。

(3.8) を見れば、$k=2$ での後退消去はこの方策どおりである。(3.2) で未定義だった $C^{(k)}_{i,j}$ $(i \le k)$ が定義されていることに留意しつつ、$k=3$ での後退消去を再考する。Row(1) の第 $j$ 要素は Row(3) による消去で $(C^{(2)}_{3,3}C^{(2)}_{1,j} - C^{(2)}_{1,3}C^{(2)}_{3,j})/C^{(1)}_{2,2}$ となり、公式 (3.6) により $C^{(1)}_{i',j'}$ $(i'=2,3,1;\ j'=2,3,j)$ を要素とする 3×3 行列式/$[c_{1,1}]^2$ で表される。$k>3$ では

$$C^{(k)}_{i,j} = (C^{(k-1)}_{k,k}C^{(k-1)}_{i,j} - C^{(k-1)}_{i,k}C^{(k-1)}_{k,j})/C^{(k-2)}_{k-1,k-1} \qquad (i<k-1,\ k<j) \tag{3.12}$$

となり、同様に 3×3 行列式で表される。$C^{(3)}_{2,j}$ は扱いが異なる。関係式

$$C^{(2)}_{2,j'} = \begin{vmatrix} C^{(1)}_{2,2} & C^{(1)}_{2,j'} \\ -1 & \end{vmatrix}, \quad C^{(2)}_{i'\ge 3,j'} = \begin{vmatrix} C^{(1)}_{2,2} & C^{(1)}_{2,j'} \\ C^{(1)}_{i',2} & C^{(1)}_{i',j'} \end{vmatrix} /c_{1,1},$$

を用いて $C^{(3)}_{2,j}$ $(j \ge 4)$ を次のように書き換える。

$$C^{(3)}_{2,j} \stackrel{\text{def}}{=} \begin{vmatrix} C^{(2)}_{3,3} & C^{(2)}_{3,j} \\ C^{(2)}_{2,3} & C^{(2)}_{2,j} \end{vmatrix} /C^{(1)}_{2,2} = \begin{vmatrix} C^{(1)}_{2,2} & C^{(1)}_{2,3} & C^{(1)}_{2,j} \\ C^{(1)}_{3,2} & C^{(1)}_{3,3} & C^{(1)}_{3,j} \\ -1 & & \end{vmatrix} /c_{1,1} = \begin{vmatrix} C^{(1)}_{3,3} & C^{(1)}_{3,j} \\ C^{(1)}_{2,3} & C^{(1)}_{2,j} \end{vmatrix} /c_{1,1}.$$

上式右辺に公式 (3.6) を適用すれば、$C^{(3)}_{2,j}$ は $C^{(0)}_{i',j'}$ $(i'=1,2,3;\ j'=1,3,j)$ を要素とする 3×3 行列式で表される。同様に、$C^{(k)}_{k-1,j}$ $(k \ge 4)$ に対しては

$$C^{(k)}_{k-1,j} = (C^{(k-2)}_{k,k}C^{(k-2)}_{k-1,j} - C^{(k-2)}_{k-1,k}C^{(k-2)}_{k,j})/C^{(k-3)}_{k-2,k-2} \tag{3.13}$$

なので、$C^{(k-3)}_{i',j'}$ $(i'=k-2,k-1,k;\ j'=k-2,k,j)$ を要素とする 3×3 行列式で表される。公式 (3.12) と (3.13) では、$C$ の肩の指数と分母因子が異なることに注意されたい。

以上より、後退消去 $(k \ge 3)$ での不完全な誤差回避策が得られたが、公式 (3.12),(3.13) およびそれらを 3×3 行列式に変換したものはそれぞれ異なる分母因子を持ち、分母因子が相対的に小さい場合には使ってはならない。不完全とはそういう意味である。

## 4 数値消去後のパラメータ要素の線形方程式系の解法

本章では数値消去後の線形方程式系の対角化を考える；実際には対角化は行わず、最後の公式だけを使用する。数値消去後の線形方程式系は下記の左辺の形である；簡単のため $C^{(n)}_{i,j}$ を $e_{i,j}$ $(1 \le i \le m;\ n+1 \le j \le m+1)$ と表す。さらに、行列の上部の $n$ 行を下部に移動し、$s=m-n$ とおいて、$e_{n+i,n+j}$ を $e'_{i,j}$ $(1 \le i \le s;\ 1 \le j \le s+1)$ と表す。

$$\begin{pmatrix} C^{(n)}_{1,1} & & & e_{1,n+1} & \cdots & e_{1,m+1} \\ & \ddots & & \vdots & \ddots & \vdots \\ & & C^{(n)}_{n,n} & e_{n,n+1} & \cdots & e_{n,m+1} \\ & & & e_{n+1,n+1} & \cdots & e_{n+1,m+1} \\ & & & \vdots & \ddots & \vdots \\ & & & e_{m,n+1} & \cdots & e_{m,m+1} \end{pmatrix} \Rightarrow E = \begin{pmatrix} & & & e'_{1,1} & \cdots & e'_{1,s+1} \\ & & & \vdots & \ddots & \vdots \\ & & & e'_{s,1} & \cdots & e'_{s,s+1} \\ C^{(n)}_{1,1} & & & e_{1,n+1} & \cdots & e_{1,m+1} \\ & \ddots & & \vdots & \ddots & \vdots \\ & & C^{(n)}_{n,n} & e_{n,n+1} & \cdots & e_{n,m+1} \end{pmatrix}, \tag{4.1}$$

ここで、$C_{i,i}^{(n)} = C_{n,n}^{(n-1)}$ $(1 \leq i \leq n)$ である。また、下記の行列式を $D$ とする。

$$D = \begin{vmatrix} e'_{1,1} & \cdots & e'_{1,s+1} \\ \vdots & \ddots & \vdots \\ e'_{s,1} & \cdots & e'_{s,s+1} \end{vmatrix}. \tag{4.2}$$

## 4.1 記号行列の消去における行列式公式

行列 $E$ の対角化を次の2ステップに分けて実行する。
Step-1: Row(1), $\cdots$, Row($s$) による前進消去；$E$ の下部の $n$ 行も消去する。
Step-2: 右上部の $s \times (s+1)$ 行列の後退消去：Row(1) $\sim$ Row($s$) に対してのみ。

Step-1 は簡単である。$k$ 段目の前進消去後の公式 (3.2) より、前進消去を終了した時点での行列の第 $i$ 行は次式となる；第 $i$ 行は $i$ 段目の消去後は不変である。

$$\left( \cdots, 0, \begin{vmatrix} e'_{1,1} & \cdots & e'_{1,i} \\ \vdots & \ddots & \vdots \\ e'_{i,1} & \cdots & e'_{i,i} \end{vmatrix}, \cdots, \begin{vmatrix} e'_{1,1} & \cdots & e'_{1,i-1} & e'_{1,s+1} \\ \vdots & \ddots & \vdots & \vdots \\ e'_{i,1} & \cdots & e'_{i,i-1} & e'_{i,s+1} \end{vmatrix} \right). \tag{4.3}$$

$i \geq s+1$ のとき、$E$ の Row($s+i$) の Row(1) による前進消去は次のように行われる。

$$e'_{1,1} \times \mathrm{Row}(s+i) - e_{i,n+1} \times \mathrm{Row}(1) =$$
$$\left( \cdots, 0, e'_{1,1} C_{i,i}^{(n)}, 0, \cdots, \begin{vmatrix} e'_{1,1} & e'_{1,2} \\ e_{i,n+1} & e_{i,n+2} \end{vmatrix}, \cdots \right).$$

帰納法により、最後の消去後の Row($i$) は次式となる；$C_{i,i}^{(n)} D$ は第 $i$ 要素である。

$$\left( \cdots, 0, C_{i,i}^{(n)} D, 0, \cdots, \begin{vmatrix} e'_{1,1} & \cdots & e'_{1,s} & e'_{1,s+1} \\ \vdots & \ddots & \vdots & \vdots \\ e'_{s,1} & \cdots & e'_{s,s} & e'_{s,s+1} \\ e_{i,n+1} & \cdots & e_{i,n+s} & e_{i,m+1} \end{vmatrix} \right). \tag{4.4}$$

Step-2 は前章の後退消去と同じように行う。たとえば第2段目の前進消去後、$E$ の Row(1) を次のように書き換える。

$$\left( e'_{1,1}, \begin{vmatrix} e'_{1,1} & e'_{1,2} \\ -1 & \end{vmatrix}, \begin{vmatrix} e'_{1,1} & e'_{1,3} \\ -1 & \end{vmatrix}, \cdots \right).$$

すると、Row(2) による消去で Row(1) は次式になる。

$$\left( \begin{vmatrix} e'_{1,1} & e'_{1,2} \\ e'_{2,1} & e'_{2,2} \end{vmatrix}, 0, \begin{vmatrix} e'_{1,1} & e'_{1,2} & e'_{1,3} \\ e'_{2,1} & e'_{2,2} & e'_{2,3} \\ -1 & & \end{vmatrix}, \cdots \right).$$

同様に行うと、対角化終了直後の行列の第 $i$ 行は次式となる；下記の $D$ は第 $i$ 要素で、右端の行列式の $-1$ は第 $i$ 列に位置する。これより Cramer の公式が得られる。

$$\left(\cdots,\ 0,\ D,\ 0,\cdots,\ \begin{vmatrix} e'_{1,1} & \cdots & e'_{1,s} & e'_{1,s+1} \\ \vdots & \ddots & \vdots & \vdots \\ e'_{s,1} & \cdots & e'_{s,s} & e'_{s,s+1} \\ & & -1 & \end{vmatrix}\right). \tag{4.5}$$

$D_j$ $(1 \le j \le s)$ は次の $s \times s$ 行列式とする；$(e'_{1,s+1}, \cdots, e'_{s,s+1})^t$ は第 $j$ 列である。

$$D_j = \begin{vmatrix} e'_{1,1} & \cdots & e'_{1,s+1} & \cdots & e'_{1,s} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ e'_{s,1} & \cdots & e'_{s,s+1} & \cdots & e'_{s,s} \end{vmatrix}, \tag{4.6}$$

すると、(4.4) と (4.5) より、$x_1, \ldots, x_m$ の解が次式で与えられる。

$$\begin{cases} \text{for } i \le n, \quad x_i = \left(e_{i,m+1}D - \displaystyle\sum_{j=1}^{s} e_{i,n+j}D_j\right)/(C_{i,i}^{(n)}D), \\ \text{for } i > n, \quad x_i = D_i/D, \qquad \text{with } C_{i,i}^{(n)} = C_{n,n}^{(n-1)}. \end{cases} \tag{4.7}$$

## 4.2 分割征服小行列式展開法

第 1 章で述べたように、我々は $D$ 及び $D_1, \ldots, D_s$ を小行列式展開法で計算する。$D$ と $D_1, \ldots, D_s$ の行列は互いに似ているので、簡単なアイデアで行列式計算を効率化できる。$D$, $D_j$ の行列をそれぞれ $E'$, $E'_j$ とし、$E'_j$ (及び $E'$) を次式のように分割する；ここで、$s_{\mathrm{L}} = [(s+1)/2]$, $s_{\mathrm{R}} = [s/2]$ である。

$$E'_j = \left(\begin{array}{c|c} L_j & R_j \end{array}\right), \quad \begin{cases} L_j : \text{left } s_{\mathrm{L}} \text{ columns}, \\ R_j : \text{right } s_{\mathrm{R}} \text{ columns}. \end{cases} \tag{4.8}$$

もし $j \le s_{\mathrm{L}}$ (resp., $j > s_{\mathrm{L}}$) ならば $R_j$ (resp., $L_j$) は $j$ によらない。したがって、行列式を上記の分割で Laplace 展開法で計算するなら、$D$ と $D_1, \ldots, D_s$ 全体で同じ小行列式の計算を避けることができる。$I$ と $I'$ は自然数の集合で、$I \cup I' = \{1, 2, \ldots, s\}$ かつ $\#(I) = s_{\mathrm{L}}$, $\#(I') = s_{\mathrm{R}}$ を満たすとする。このとき、$D_j$ は次の Laplace の展開公式で計算できる。

$$D_j = |E'_j| = \sum_I \mathrm{sgn}(I)\, |L_{j,I}| \times |R_{j,I'}|. \tag{4.9}$$

ここで、$\sum_I$ は異なる $I$ 全体に渡る和であり、$L_{j,I}$ (resp., $R_{j,I'}$) は縦長行列 $L_j$ (resp., $R_j$) の中から $I$ (resp., $I'$) で指定される $s_{\mathrm{L}}$ 行 (resp., $s_{\mathrm{R}}$ 行) を取り出して作られる部分行列である。また、$\mathrm{sgn}(I) = (-1)^{\sum_{i=1}^{s_{\mathrm{L}}}(i+I_i)}$、ただし $I_i$ は $I$ の第 $i$ 要素、である。

# 5 種々の注釈、特に異種の誤差低減法

本稿では、数値消去中の行列要素の行列式表現と、その表現に基づく (不完全な) 誤差低減法を記述した。しかし、国際会議論文 [14] に記した幾つかの重要な事柄を省略したので、以下に注釈として簡単に記述する。

**注1** 誤差低減は、多くの場合、ピボッティングで十分だと思うが、3.1 節で述べたようにピボットは注意して選ぶ必要がある。多くの場合、数値列だけからピボットを選んでよいだろうが、数値列から選べば記号行列が悪条件となり、記号要素を考慮して選べば数値消去が桁落ちをひき起こす、そんな場合もあり得る。したがって、異質な誤差低減法も用意する必要がある (**注4** で一つの方法を簡単に記述する)。なお、3×3 行列式に変換して計算する方法はピボッティングなしで使えるが、ピボットが小さいときに一段の消去に使うにだけにすべきで、連続して使うのは良くない。

**注2** 本稿では数学的観点から消去における行列式表現を導出したので、必然的に (3.2) や (3.6) の公式のように分母因子も導出した。しかし、(3.12) と (3.13) が示すように、分母因子の現れ方は一様でない。しかも、分母因子を計算することは行列式表現をそのまま計算することであり、消去が進行すると共に行列要素ノルムが指数関数的に増大または減少する場合がほとんどである。ゆえに、係数行列は各列の消去後に規格化するべきである。規格化する場合、たとえば (3.2) 公式は下記でよい。

$$C_{i,j}^{(k)} = C_{i,j}^{(k-1)} - (C_{i,k}^{(k-1)}/C_{k,k}^{(k-1)}) \times C_{k,j}^{(k-1)}. \tag{5.1}$$

なお、たとえば $c_{i,1} = 0$ $(i \geq 2)$ ならば、この要素を含む第 $i$ 行には消去用の演算を施す必要はないが、規格化しない場合には施す必要がある。

**注3** Bareiss の 2 ステップ算法では公式 (3.6) が用いられる。我々の誤差回避用の算法は Bareiss の 2 ステップ算法と同じと思われるかもしれないが、若干異なる。Bareiss の算法は公式 (3.6) を用い、$C_{i',j'}^{(0)}$ から $C_{i,j}^{(2)}$ を、$C_{i',j'}^{(2)}$ から $C_{i,j}^{(4)}$ を、..., と計算する ($n$ が偶数の場合には最後の $C_{i',j'}^{(n-2)}$ から $C_{i,j}^{(n-1)}$ の計算で公式 (3.2) を用いる)。次に後退消去を下記のように行って対角化する。前進消去直後は、Row(1) と Row(2) は初期行列のままである。そこで、公式 (3.2) を用いて Row(2) を Row(1) で消去し、ついで Row(1) を消去後の Row(2) で消去する。それ以降の後退消去も同様である。本稿の式で言うと公式 (3.13) で計算している。これを見ると、前進消去では誤差が抑えられているが、後退消去では、たとえば Row(1) を Row(2) で消去するときに大きな桁落ち誤差が発生する可能性がある。

**注4** ピボッティングとは全く異質の、独立記号を用いた誤差低減法を簡単に説明する；詳細は [14] を参照されたい。3.1 節によると、桁落ち誤差は巨大な要素同士がキャンセルすることで発生するが、キャンセルする前に生き残る部分に加え込まれることが必要である。そこで、消去に用いる Row($k$) の先頭要素が小さい場合、その先頭

要素に独立記号 $s$ を掛け、消去を２段行う (２段目に項が組織的にキャンセルする)。3.1節によると、キャンセルするのは $s$ に比例しない項なので、$s$ に比例する項だけを取り出し、$s=1$ としたものを消去後の要素とすればよい。

**注5** 注４に述べた誤差低減策や分割征服小行列式展開法も含め、いくつかの数値実験の結果を [14] に記載したので参照されたい。注４で説明した誤差回避策は数値消去の際に要素に記号処理を行うので、計算時間がかかることが気になるが、数値実験の結果をみれば、大したことはないことが分るだろう。

## 6 まとめと今後の課題

物理法則では定数は記号で表されるので気が付かないが、産業分野では大きさが非常に異なる数値が混用される。したがって、それらの記号の一部に実際の数値を代入した線形方程式系の係数は、非常に広い範囲に分布するだろう。本稿では、ピボッティング以外に二つの異質の誤差低減策を考案したので、多分、産業分野の計算にも対処できると思う。しかしながら、産業分野の実例で試した訳ではない。今後は実際計算を扱いながら算法を精錬することが必要である。

本稿では『微分消去における誤差低減策』を省略したが、それは前章の注４で説明した独立記号を用いる方法と非常に似ている。この方法は、微分消去のみならず、有理式体上のグレブナー基底の計算でも有用であろうと思っている。この計算では、簡約後の多項式の有理式係数が膨張するので簡単化が必要だが、この計算で発生する係数膨張メカニズムに対して我々の方法が対処できるからである。

実は、“モデルに基づく開発” では、大規模疎行列に特有の課題の方が誤差低減化よりも重要である。重要な課題は、1) 線形方程式の解はパラメータに関する有理式となるが、パラメータの個数が多く変数が多い場合には単純に計算すると膨大な有理式が得られて始末に困るので、パラメトリック解を簡潔で扱い易い形で得ること、2) 線形方程式全体を一つの系として見るのではなく、全体を簡潔に表す単純化された系や、局所的に重要な部分系などを自動的に作成すること、などである。これらの課題の解決を目指した研究については [8] を参照されたい。

謝辞　本研究は日本学術振興会・科学研究費 (23500003) で支援された。

## 参 考 文 献

[1] E.H. Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. Math. Comp. **22** (1968), 565-578.

[2] F.E. Cellier and E. Kofman. *Continuous System Simulation*, Chap. 7: Differential Algebraic Equations, Springer-Verlag 2006.

[3] N.J. Higham. *Accuracy and Stability of Numerical Algorithm.* SIAM, Philadelphia, 1995.

[4] S.Y. Ku and R.J. Adler. Computing polynomial resultants: Bezout's determinant vs. Collins' reduced P.R.S. algorithm. Communi. ACM, **12** (1969), 23-30.

[5] E. Mansfield. Differential Gröbner bases. Ph.D Thesis (University of Sydney, Australia), 1991.

[6] T. Sasaki. A practical method for floating-point Gröbner basis computation. *Proceedings of Joint Conf. of ASCM 2009 and MACIS 2009: COE Lecture Note 22*, 167-176, Kyushu Univ., 2009.

[7] T. Sasaki: A theory and an algorithm of approximate Gröbner bases. *Proceedings of SYNASC2011 (Symbolic and Numeric Algorithms for Scientific Computing)*, West University of Timisoara, Romania: SYNASC 2011, IEEE Computer Society, 23-30, 2012.

[8] T. Sasaki, D. Inaba and F. Kako. Solving parametric sparse linear systems by local blocking (in preparation).

[9] T. Sasaki and F. Kako. Computing floating-point Gröbner base stably. *Proceedings of SNC2007 (Symbolic Numeric Computation)*, 180-189, London, Canada, 2007.

[10] T. Sasaki and F. Kako. Floating-point Gröbner basis computation with ill-conditionedness estimation. *Proceedings of ASCM2007 (Asian Symposium on Computer Mathematics)*: LNAI 5081, 278-292, Springer, 2008.

[11] T. Sasaki and F. Kako. Term cancellations in computing floating-point Gröbner bases. *Proceedings of CASC2010 (Computer Algebra in Scientific Computing)*: LNCS 6244, 220-231, Springer, 2010.

[12] T. Sasaki and H. Murao, "Efficient Gaussian elimination method for symbolic determinants and linear systems", ACM Trans. Math. Software, **8** (1982), 277-289.

[13] T. Sasaki and T. Sato. Cancellation Errors in Multivariate Resultant Computation with Floating-point Numbers. ACM SIGSAM Bulletin **32** (1998), 12-20.

[14] T. Sasaki and T. Yamaguchi. On algebraic preprocessing of floating-point DAEs for numerical model simulation. *Proceedings of SYNASC2013 (Symbolic and Numeric Algorithms on Scientific Computing*, West University of Timisoara, Romania: SYNASC2013, 81-88, IEEE Computer Society, 2014.