数理解析研究所講究録
第1955巻 2015年 62-75

パラメータ係数の疎な線形方程式系の局所ブロック化による解法

Solving Parametric Sparse Linear Systems by Local Blocking

佐々木 建昭 (Tateaki Sasaki) *
筑波大学 名誉教授
(UNIVERSITY OF TSUKUBA)

稲葉 大樹 (Daiju Inaba) †
(公財) 日本数学検定協会
(JAP. ASSOC. MATH. CERTIFICATION)

加古 富志雄 (Fujio Kako) ‡
奈良女子大学 理学部
(NARA WOMEN'S UNIVERSITY)

Abstract

数値係数の疎な線形方程式系に対してはブロック三角化という非常に有効な方法が1950年代から知られている。その各ブロックは係数行列から作られる有向グラフの各強連結成分 (maximal) に対応する。パラメータ係数の場合、ブロックは数式処理するには大きすぎることが多いので、本稿では局所ブロックの概念を提案する。局所ブロックとはグラフの強連結成分中の non-maximal な強連結部分グラフと定義する。局所ブロックの中では、ブロック外の変数を外部変数と扱って、ブロック内の変数について解くことができる。この局所解法により、一部の解をパラメータ空間内で縮退させる (パラメータ間の) 特別な関係を求めたり、パラメータに関する部分式を "システムパラメータ" で組織的に置き替えることにより、解を簡潔な形で表現することができる。しかし、最も重要なことは "特徴系" と命名した部分系を求めることにより、大規模系を小さな系に分割し、大規模系の分割征服 (divide-and-conquer) に道を拓くことであろう。本稿では、大規模系としては産業界の Model-Based Development で扱われるような系を念頭に、特徴系や局所系を如何に計算するかについて詳述する。

1 はじめに

産業界では近年、複雑な機械やプラントの開発において、"**モデルに基づく開発** (Model-Based Development$\cdots$MBD)" なる開発法が盛んになっている。モデルでは、機械やプラ

*sasaki@math.tsukuba.ac
†d.inaba@su-gaku.net
‡kako@ics.nara-wu.ac.jp

ントの動きをNewtonの運動法則や電磁気学の法則に基づき微分方程式で記述し、機械やプラントの構造を座標とパラメータに関する代数方程式で記述する。すなわち、モデルは多数のパラメータを含む大規模な**連立微分代数方程式**である。各方程式は大抵、数個の項からなり、**非常に疎である**のが特徴である。最終的な目的は、微分代数方程式系のパラメータ解を用いて、パラメータの最適値を求めることである。産業界の現状は、微分代数方程式系を数式処理で数値計算用に変換したのち、パラメータに数値を代入して数値解を求め、解の挙動をフィードバックしつつパラメータを最適値に近づけている [1]。しかし、数式処理で各種のパラメータ解析を行いたいとの希望もある。

著者の一人 (T.S.) は昨年の数理研研究集会で、Yamaguchi とともに「パラメータ入りの連立線形方程式の誤差低減法」と題して、多数のパラメータを含む浮動小数係数の線形方程式系を如何に誤差を少なく解くか、について発表した [10, 9]。最終的にはパラメータ係数の連立微分代数方程式を扱いたいのだが、研究は初歩段階なので、単なる線形方程式系に限定したのである。本稿では、パラメータ係数の連立線形方程式は産業界で扱われるようなものを念頭に、パラメータの数式処理的決定に資するような解法を追求する。

疎な連立線形方程式に関しては、**ブロック三角化**という有名な技法が1950年代に開発されている [3, 5]。第2章で詳述するが、疎な線形方程式系の係数行列はブロック三角化でき (上三角でも下三角でもよい)、下三角化なら上のブロックから順に解ける。しかも、各方程式間の依存関係をある種の**有向グラフ**で表すと、各ブロックはグラフの**強連結成分** (strongly connected component $\cdots$ **SCC** と略記) に対応する。そして、全SCCを高速に計算する算法がTarjanにより得られている [11, 2, 6]；計算量はグラフの (頂点数+辺数) に比例する。係数行列を (パラメータに適当な数値を代入した) 数値行列として扱う限り、筆者らの出番はない。だが、係数行列をパラメータ込みで扱うと事態は一変する。多くの場合、ブロックは数式処理するには大きすぎるのである。そこで、筆者らは**局所ブロック** (local block) の概念を提案する。SCCがmaximalな強連結部分グラフであるのに対して、局所ブロックとはnon-maximalな**強連結部分グラフ** (SCsubGと略記する) であると定義する。第3章で詳述するが、強連結グラフに拘る理由は、対応する連立方程式が**形式的に解ける** (formally solvable) からである。

局所ブロックが求まれば、そのブロック外の変数を "外部変数" と扱い、局所ブロック内の変数について解くことができる。したがって、a) 解の性質を大きく変えるパラメータ間の関係式を求めること、b) パラメータの部分式を "システムパラメータ" で置き換えて解全体を簡潔に表すこと、が可能になる。さらに、c) 系全体を簡潔に表現する "特徴系" を決めることができるだろう。なお、特徴系は第4章で定義し議論する。

第2章では、ブロック三角化と局所ブロックについて簡単に説明するとともに、上記のa) と b) を具体的に例で説明する。第3章では、係数行列と有向グラフの対応付けと取り扱い方 (well-known) を解説し、[8] で採用した局所ブロック計算法を説明して、その欠点を指摘する。第4章では、系全体の "分割征服的" なパラメータ解析に資するべく特徴系を定義して、特徴系を先に決め、その特徴系を用いて局所ブロックを決める簡潔な方法を記述する。第5章では、第4章の局所ブロック化を頂点数100のグラフで実験した結果を述べる。なお、本稿は本年度に発表した論文 [8] と [7] を統合したものである。

2 ブロック三角化と局所ブロックの簡単な説明

本稿では、$\boldsymbol{x}$ と $\boldsymbol{p}$ はそれぞれ変数全体とパラメータ全体を表し、$A \in \mathbb{C}[\boldsymbol{p}]^{l' \times l}, B \in \mathbb{C}[\boldsymbol{p}]^{m \times m}, L \in \mathbb{C}[\boldsymbol{p}]^{n \times n}$ は与えられた線形方程式系の係数行列、三角化されたブロックの係数行列、局所ブロックの係数行列、をそれぞれ表すものとする。また、各行列の第 i 行に対応する方程式を Eqi と表す。

図 1 は、行列 A のブロック上三角化を単純化して図示したものである。左上部の横長ブロックは過少決定系 (変数数 > 方程式数) 全体を表している。右下部の縦長ブロックは過剰決定系 (変数数 < 方程式数) 全体を表している。中の二つの正方行列が主座ブロックである。下のブロックから順に解くならば、解いたブロックの解を上のブロックの右辺に代入することにより、各ブロックは他とは独立に解けることが分かる。かくして、系全体の分割解法が得られる。さらに、可解性は各ブロックの可解性に帰着する。

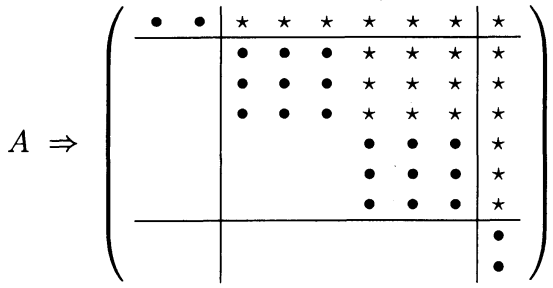

図 1 : ブロック上三角化 ($\star$ はブロック要素とはみなさない)

次に、局所ブロックとその利用法を簡単な例で説明する。

例 1 a, b, c, d, f, g をパラメータとする次の線形系 $B\boldsymbol{x} = \boldsymbol{c}$ を考えよう。

$$B = \begin{pmatrix} a & b & 0 & 0 & 0 \\ -b & 2a & 0 & 0 & 1 \\ 0 & 0 & c & d & 0 \\ 0 & 0 & d & 2c & 2 \\ 0 & f & 0 & g & 3 \end{pmatrix}, \quad \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}, \quad \boldsymbol{c} = \begin{pmatrix} 3 \\ 2 \\ 1 \\ -1 \\ -2 \end{pmatrix}. \tag{2.1}$$

$D_1 = \begin{vmatrix} a & b \\ -b & 2a \end{vmatrix}$, $D_2 = \begin{vmatrix} c & d \\ d & 2c \end{vmatrix}$ とし、Eq1 と Eq2 を連立して x_1 と x_2 について解き、同様に Eq3 と Eq4 を連立して x_3 と x_4 について解くと、

$$x_1 = (6a - 2b + bx_5)/D_1, \qquad x_2 = (2a + 3b - ax_5)/D_1,$$
$$x_3 = (2c + d + 2dx_5)/D_2, \qquad x_4 = -(c + d + 2cx_5)/D_2.$$

これらの "局所解" を Eq5 に代入すると、次式を得る。

$$x_5 = \frac{-2D_1D_2 + g(c+d)D_1 - f(2a+3b)D_2}{3D_1D_2 - 2cgD_1 - afD_2}. \tag{2.2}$$

最後に、x_5 を上々式に代入すると $x_1, \ldots, x_4$ が定まる。部分行列 $\begin{pmatrix} a & b \\ -b & 2a \end{pmatrix}$ と $\begin{pmatrix} c & d \\ d & 2c \end{pmatrix}$ が B の局所ブロックである。

$D_1 = 0$ の場合を考えよう。系 {Eq1, Eq2} ($a \neq 0$) は次の系に等価である。

$$\{ax_1 + bx_2 = 3,\ D_1 x_2 = 3b + 2a - ax_5\} \quad (\text{so long as } a \neq 0).$$

したがって、$D_1 = 0$ なら $x_5 = (2a+3b)/a$ となり、x_5 の解はパラメータ空間で縮退することがわかる。すなわち、解の性質を大きく変える "パラメータ間の関係式" を、全部ではないが、かなり細かく知ることができる。また、D_1 や D_2、さらに (2.2) の右辺の分子・分母を "システムパラメータ" で置き換えることにより、通常は大きな有理式になる解を (システムパラメータに関する "入れ子表現" として) 簡潔に表すことができる。 □

3 グラフ理論による定式化と [8] での局所ブロック化

ブロック三角化は有向グラフを用いて定式化され実行される。グラフは**頂点**と**辺**で構成される。頂点 v から w へ向かう辺を $(v \to w)$ と表す。行列 A は以下の手順でグラフに変換される；簡単のため $\min\{l, l'\} = l$ とする。対角要素 $a_{11}, a_{22}, \ldots, a_{ll}$ を頂点 $v_1, v_2, \ldots, v_l$ にそれぞれ対応させ、非零の非対角要素 a_{ij} $(1 \leq i \neq j \leq l)$ を辺 $(v_j \to v_i)$ に対応させる (詳しく言うと非零要素が係数行列の対角線上にくるように "最大マッチング" 操作 [12] で方程式を入れ替える必要がある)。上記の手順で得られたグラフは行列 A の**関連グラフ**と呼ばれ、以下では G_A と表すことにする。

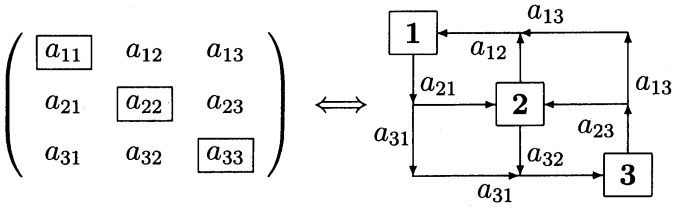

図2： 行列 A と関連グラフ G_A との対応

有向グラフ S は、S のどの頂点も他の任意の頂点から到達できるとき**強連結**という。G の部分グラフ S は、強連結であり、かつ S の外部にある G のどの頂点や辺を取り込んでも強連結でなくなるという意味で maximal であるとき、**強連結成分**といい SCC と略記する。SCC はその内部に強連結部分グラフを含み得る。G 内の頂点 $v_1, v_2, \ldots, v_n$ と $n-1$ 個の辺 $(v_1 \to v_2)$, $(v_2 \to v_3)$, $\ldots$, $(v_{n-1} \to v_n)$ からなる部分グラフ P を**路**といい、$P = (v_1, v_2, \ldots, v_n)$ と表す。$v_1 = v_n$ のとき P は**周回路**という。

定義 1 (形式的可解性；グラフは行列要素を "零 or 非零" としか扱わないから必要) 系 S: $L\boldsymbol{x} = \boldsymbol{c}$ は $\boldsymbol{x}$ に関する線形方程式系とする。ここで、$\boldsymbol{c}$ は $\boldsymbol{x}$ の要素を含まないが、それ以外の変数は含んでよい。行列 L の各非零要素を独立なパラメータとみなすとき S が x について可解ならば、S は形式的に可解であるという。 □

定理 1 (強連結性と形式的可解性を結びつける定理：既知)
L は $\mathbb{C}[\boldsymbol{p}]$ 上の $n \times n$ 行列で G_L はその関連グラフ関連グラフとする。G_L が強連結ならば線形方程式系 $S : L\boldsymbol{x} = \boldsymbol{c}$ は形式的に可解である。

証明 L は方程式系ゆえ $n \geq 2$ である。強連結性より G_L の任意の頂点たちを結ぶ周回路があるので、それを $P = (v_{i_1}, \ldots, v_{i_\nu}, v_{i_1})$ とする。路 P を歩くことは S の項を辿ることに対応するので、辿る項を順に $b_{i_1 i_1} x_{i_1} \to b_{i_2 i_1} x_{i_1} \to \cdots \to b_{i_\nu i_\nu} x_{i_\nu} \to b_{i_1 i_\nu} x_{i_\nu} \to b_{i_1 i_1} x_{i_1}$ とする。これより、P に現れる任意の変数 x_j は少なくとも S の二つの方程式に現れ、S の各方程式は少なくとも二つの非零項を持つことが分かる。したがって、S に対して形式的にガウス消去法を適用できるので、S は形式的に可解である。 □

グラフ演算を計算機で如何に実行するかを説明する。グラフの頂点同士の結び付きは**隣接リスト**で表すことが多い。グラフ G_B の頂点が $v_1, v_2, \ldots, v_m$ ならば、サイズ m の配列を用意し、v_i $(1 \leq i \leq m)$ から出ていく辺が $(v_i \to v_{j_1}), \ldots, (v_i \to v_{j_t})$ であれば、$\mathrm{Adj}[v_i] := (v_{j_1}, \ldots, v_{j_t})$ とする。G_B の全頂点を規則的に辿る主な方法は**深さ優先探索 (DFS)** と**横幅優先探索 (BFS)** である。どちらも探索を開始する頂点 v_1 を (勝手に) 選ぶ。その後の操作を DFS から説明する。$\mathrm{Adj}[v_1] = (v_{i_1}, \ldots, v_{i_s})$ とすれば、DFS はまず辺 $(v_1 \to v_{i_1})$ を歩き、v_{i_1} を $\mathrm{Adj}[v_1]$ から消去する。次に、$\mathrm{Adj}[v_{i_1}] = (v_{j_1}, \ldots, v_{j_t})$ とすれば、DFS は辺 $(v_{i_1} \to v_{j_1})$ を歩き、v_{j_1} を $\mathrm{Adj}[v_{i_1}]$ から消去する。v_{j_1} が既に辿った頂点なら v_{j_2} に移る。$v_{j_1}, \ldots, v_{j_t}$ を全て辿り終えると、頂点 v_{i_2} に戻り、探索を再開する。一方、BFS はまず頂点 $v_{i_1}, \ldots, v_{i_s}$ をこの順に辿り、すべて辿り終えると、$v_{i_1}, \ldots, v_{i_s}$ から辺一つを歩くことで辿れる頂点をすべて辿る。そしてこれを繰り返す。

DFS/BFS が辺 $(v \to w)$ を歩くとき、もしも w が未訪問の頂点なら $(v \to w)$ を**前進辺**といい、そうでなければ**後退辺**という。DFS/BFS は、後退辺を歩くや直ちに元の頂点 v に引き返す。このとき、DFS/BDF は**逆路**を辿るという。頂点 v_{i_1} から $v_{j_1}, \ldots, v_{j_t}$ すべてを辿ったあと、v_{i_1} に戻るときも逆路を辿るという。

例 2 DFS/BFS がどのようにグラフを辿るか、次の例でみよう。

$$\mathrm{Adj} = [(2,8), (3,5), (2,4), (1), (6,7), (3), (4), (9), (8,7).] \tag{3.1}$$

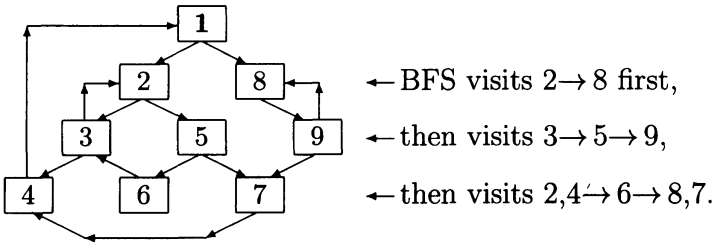

図 3 ： DFS(左) と BFS(右) との比較

頂点 1 から出発すると、DFS は前進辺でみれば各頂点を $1 \to 2 \to \cdots \to 8 \to 9$ と辿る。詳しくいうと次のように辿る；ここで、後退辺で辿る頂点 v を $\langle v \rangle$、逆路で辿る頂点 v を

(v) と表す。

$$1\to 2\to 3\to \langle 2\rangle\to (3)\to 4\to \langle 1\rangle\to (4)\to (3)\to (2)\to 5\to 6\to \langle 3\rangle\to (6)\to (5)\to$$
$$7\to \langle 4\rangle\to (7)\to (5)\to (2)\to (1)\to 8\to 9\to \langle 8\rangle\to (9)\to \langle 7\rangle\to (9)\to (8)\to (1).$$

一方、頂点 1 から出発した BFS は、各頂点を次のように辿る。

$$1\to 2\to 8\to 3\to 5\to 9\to \langle 2\rangle\to 4\to 6\to 7\to \langle 8\rangle\to \langle 1\rangle\to \langle 3\rangle\to \langle 4\rangle.$$ □

全ての SCC を高速に計算する算法で Tarjan は DFS を採用した。強連結部分グラフを見つけるには各頂点を路に沿って辿ることが必然であり、しかも後退路が周回路の目印となる DFS の方が便利だったのである。論文 [8] で筆者らも DFS を用いた。しかし、我々の目的にかなう SCsubG(強連結部分グラフ) を選び出すには不便だった。一般に SCsubG は非常に多数あるので、すべての SCsubG を求めた後で欲しい SCsubG を選び出すのはたまらない。一方、我々の欲しい SCsubG は連立方程式の解法に有用であるべきで、そのためには既に見つけた SCsubG を中に含むようなものが欲しい。そこで筆者らは、目的にかなう SCsubG が見つかると、“大頂点” に縮約することにした。このアイデアはいいと思ったが、大頂点を含むグラフの包含関係のチェックが複雑になった。DFS が辿る辺は、大頂点に縮約した SCsubG 内の頂点にも容赦なく繋がるからである。さらに、特徴系は求めた SCsubG を統合して簡単に決まるだろうと思っていたが、全くそうではなかった。そこで、次章で述べるように、考え方を逆転することにした。

4 特徴系を先に BFS で決める：論文 [7] の紹介

実は、論文 [8] を書いている最中は SCsubG の計算法に気をとられ、特徴系については計算法のみか利用法もよく考えてはいなかった。SCsubG の計算の複雑さに辟易し、何とか SCsubG の簡単な計算法はないかと考えるうちに、特徴系の利用法も考えて、特徴系を先に構成するアイデアに行き着いた。特徴系はどう利用すべきだろうか？ 研究の最終の目的は大規模な微分代数方程式系をパラメータのままで処理することである。数式処理で簡単に扱える系は研究する価値はないので、系は直接的に扱うには荷が重すぎるとして、そんな系を扱う方法を開発したい。そこで、大規模系をもう少し詳しく見てみよう。

複雑な機械やプラントは、製作と保守を簡単にするためモジュール化するはずである。具体的には、全体をいくつかの bodies に分割し、各 body 内では parts が密接に結合しているが、異なる body は比較的少数個の方程式で結びついている、そんな構造をしているものが多いだろう。そうだと仮定すれば、系全体に関わって各 body を結びつける一群の方程式を特徴系とみなし、特徴系で緩く結びついている bodies を SCsubGs とみなせる。ここで、“緩い結合” とは、少数個の方程式または項で結びついているということである。以上より、特徴系はどうあるべきかが見えてくる：大規模系が多くの小規模系 (bodies) の緩い結合体であるとき、大規模系を小規模系に分割するような系を探索し、それを特徴系としよう、と。そうすれば、大規模系を “分割征服法” で攻略する道が開ける。特徴系をグラフの言葉で言うと次のように定義できるだろう。

定義 2 (特徴系：大規模系を多くの小規模系に分割するのに利用する)
系 S: $B\boldsymbol{x}=\boldsymbol{c}$, $B\in\mathbb{C}[\boldsymbol{p}]^{m\times m}$, は疎な線形方程式系で、$G_B$ は係数行列 B の関連グラフで強連結とする。S の特徴系 $S_{\rm char}$ とは S の部分系で、全体系 S を "大域的に近似し"、かつ $S_{\rm char}$ の係数行列の関連グラフ $G_{\rm char}$ が強連結で、G_B から $G_{\rm char}$ を引くことで、G_B が多くの強連結な部分グラフに分割されるものとする。 □

この定義では "S を大域的に近似する" が曖昧である。グラフでは頂点どうしの連結性だけが意味があり、ある頂点が大域的か否かなど無意味である。大域的と言うのはユーザから見て定義できる概念である。さらに、特徴系を用いて S を多くの部分グラフに分割したとき、それが有用な分割であるか否かもユーザの価値基準による。すなわち、有用な特徴系を決めるにはユーザの価値判断を何らかの形で取り込む必要がある。そこで筆者らは、特徴系を次のように計算することにした。

- 特徴系の関連グラフ $G_{\rm char}$ は周回路 P で、P の代表的な頂点 $U=(u_1,u_2,\ldots,u_k,u_1)$ はユーザにより指定されるものとする。P 自体は、指定された頂点を順に通過する**最短経路**として計算機が計算するものとする。

周回路 P の計算は、$i=1\to 2\to\cdots\to k\to 1$ に対し、u_i から u_{i+1} (ただし $u_{k+1}=u_1$) に到る最短経路 P_i を計算し、$P=(P_1,P_2,\ldots,P_k)$ とする。

上記のように特徴系を決めるとき、DFS と BFS のどちらを用いても大差ないと思える。そこで、筆者らは両方でプログラムを書いてみた。得られた結果には大差はなかったが、プログラミングには大差があった：BFS を用いる方が圧倒的に簡単なのである。両者によるプログラムの概略は論文 [7] に記述したが、本稿では BFS のみを用いる。

補題 1 与えられた頂点 u から v へ行く路を BFS で見つけるとき、後退辺を通る路は無視してよく、無視して得られた路 (複数個あり得る) は最短経路である。

証明 任意の頂点 w に対し $\mathrm{Adj}[w]=(w_1,\ldots,w_s)$ とすれば、w から各頂点 $w_1,\ldots,w_s$ への距離は 1 で最短である。BFS により u から v に行く際に後退辺 $(u'\to w)$ を通るならば、w は既に通過した頂点なのでその路は最短経路ではない。逆に、後退辺を全く通過しない路は上記のことから最短経路である。 □

たとえば、3 章例 2 の Adj に対し、$U=(1,7,4,1)$ とすれば、最短経路の組は

$$\mathrm{Ssets}=(((1,2,5,7),(1,8,9,7)),\ ((7,4)),\ ((4,1))).$$

となり、これらの経路をつなぐと、特徴系を表す周回路として次の二つが得られる。

$$\mathrm{Cycls}=((1,2,5,7,4,1),\ (1,8,9,7,4,1)).$$

一方、$U=(1,7,6,4,1)$ と設定するならば次のように周回路を二つ得る。

$$\begin{aligned}\mathrm{Ssets}=&\ (((1,2,5,7),(1,8,9,7)),((7,4,1,2,5,6)),((6,3,4)),((4,1))),\\ \mathrm{Cycls}=&\ ((1,2,5,7,4,1,2,5,6,3,4,1),\ (1,8,9,7,4,1,2,5,6,3,4,1)).\end{aligned}$$

U を少し変えただけでも特徴系は大幅に変わり得ることが分かる。グラフはどの頂点から書いてもよく、隣接リストにはどういう順番で収納してもよいので、順番を変えると全く異なって見える。U がユーザの助けなしに簡単に決まると思ってはならない。

定理 2 (DFS に基づく特徴系の計算量)
$U = (u_1, \ldots, u_k, u_1)$ が与えられたとき、*BFS*に基づく上述のプロシジャは組 $(P_1, P_2, \ldots, P_k)$ すべてを $O(\#\mathrm{edge}(G_B) \times k)$ の計算量で計算する。ここで、各 P_i $(1 \leq i \leq k)$ は u_i と u_{i+1} を結ぶ最短経路である $(u_{k+1} = u_1)$。 □

この定理の証明には上述のプロシジャの詳細が必要であるが、本稿では割愛したので証明も省略せざるをえない。興味ある読者は [7] を参照されたい。

特徴系 S_{char} が決まったとして、関連グラフ G_B を SCsubG に分割することを具体的に考えよう。S_{char} の係数行列の関連グラフを G_{char} とする。我々は、G_B を SCsubG に分割するに際して、まず G_B から G_{char} を差し引く：

$$G_{\mathrm{rem}} := G_B - G_{\mathrm{char}} \quad (G_{\mathrm{rem}}\text{は “残余グラフ”}). \tag{4.1}$$

G_{char} は周回路として計算したので、$G_{\mathrm{cycl}} = (v_1, v_2, \ldots, v_n, v_1)$ とすれば $(v_1 = u_1)$、G_{char} は n 個の頂点とそれらをつなぐ n 個の辺からなる。G_B から G_{char} を引く操作は、基本的には G_B の隣接リストから G_{char} の n 個の辺を除去することである。しかし、それでは G_B は過度に多くの SCsubG に分割される傾向があるので、引き方に若干の自由度を設ける。

G_B の隣接リストのコピー **Adj** と G_{rem} および “条件” Cond を引数として、G_{rem} の隣接リスト **Adjrem** を計算するプロシジャ *setAdjrem* を下記に示す。

```
Procedure setAdjrem (Adj,G_cycl,Cond) ==
begin  local v_1, v_2;
   lp:  v_1 := first (G_cycl);  G_cycl := rest (G_cycl);
        v_2 := first (G_cycl);
        if checkCond (Cond,v_1, v_2) = 'True then
            delete v_2 from Adj[first (G_cycl)];
   nx:  G_cycl := rest (G_cycl);
        if G_cycl = ( ) then return Adj;
        v_1 := v_2;  v_2 := first (G_cycl);
        goto lp;  end.
```

条件 Cond としては、現在のところ次の三つが用意されている。

1. Cond = 'couple：v_1 と v_2 が “couple” でない場合、Adj から辺 $(v_1 \to v_2)$ を除去する；v_1 と v_2 が couple であるとは、G_B が $(v_1 \to v_2)$ と $(v_2 \to v_1)$ を持つことである。
2. Cond = (n_1, n_2) with $n_1 < n_2$：$n_1 \leq v_1 \leq n_2$ かつ $n_1 \leq v_2 \leq n_2$ であれば、Adj から辺 $(v_1 \to v_2)$ を除去する。
3. Cond = (n_2, n_1) with $n_1 < n_2$: $v_1, v_2 \leq n_2$ または $n_1 \leq v_1, v_2$ であれば、Adj から辺 $(v_1 \to v_2)$ を除去する。

下記プロシジャ*LocBlocking*は、G_{char}の代表頂点リストUから特徴系の関連グラフG_{char}を計算し、$G_{\text{rem}} := G_B - G_{\text{char}}$をSCCsに分解して$G_{\text{rem}}$を局所ブロックに分割する。プロシジャ*findSpaths*はUから最短経路の組 $P = (P_1, P_2, \ldots, P_k)$ を計算し、*connectPaths*は$P_1, P_2, \ldots, P_k$を繋いで一本の周回路にする。*SCCdecomp*はTarjanのSCC分解算法で、*formGsys*と*formLocSyss*は与方程式系Eqsから特徴系と局所ブロックを構成する。局所ブロックがSCC分割で計算されること、得られたSCCが大きい場合には算法を再帰的に適用して再分割されることに注意されたい。

Procedure *LocBlocking*(Adj,Eqs,U, m) ==
  %% U := a tuple of vertices for G_{cycl};
begin **local** Gcyc,Gsys,Adjrem,locSs,Svtxs;
  1: Svtxs := *findSpaths*(Adj,U, m);
    Gcyc := *first*(*connectPaths*(Svtxs));
  2: Adjrem := *setAdjrem*(Adj,Gcyc,Cond);
    Svtxs := *SCCdecomp*(Adjrem);
    if Svtxs contains big SCCs **then**
      apply the algorithm **to** them;
  3: *addInEdges*(Adj,Gcyc,Svtxs);
    Gsys := *formGsys*(Gcyc,Eqs);
    locSs := *formLocSyss*(Svtxs,Eqs);
    return (Gsys,locSs); **end**;

プロシジャ*addInEdges*ついて少し説明する(*"In"*は"incoming"を意味する)。*SCCdecomp*は各SCCを結ぶ辺を除去したSCC集合を答えとするが、部分系のいくつかはそれらの辺を必要とする。たとえば下記の例2では、部分系{Eq_3,Eq_5}で(2→3)と(4→5)が必要である。上記のプロシジャで、$(v_i \to v_j)$が必要な辺なら、*addInEdges*は頂点v_jを含む一つのSCCにその辺を追加する。同じことはG_{cycl}とSCCsを結ぶ辺にも行う。

例3 上記の局所ブロックの計算法を簡単な例で見よう。
5×5行列Bは、上から $(b_{11}, 0, b_{13}, 0, 0)$, $(b_{21}, b_{22}, 0, b_{24}, 0)$, $(0, b_{32}, b_{33}, 0, b_{35})$, $(0, b_{42}, 0, b_{44}, 0)$, $(0, 0, b_{53}, b_{54}, b_{55})$, の5行からなるとする。$B$の関連グラフ$G_B$は図4の左図である。

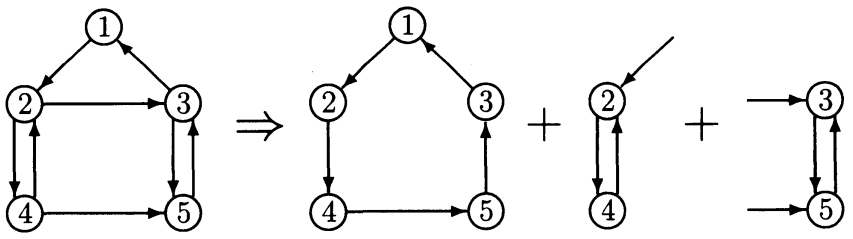

図4 : G_Bを$U = (1, 4, 5)$, Cond='couple で特徴系と局所ブロックに分解

行列 B で $b_{24}=b_{32}=b_{53}=0$ とした行列を B' とすれば、特徴系は $B'\boldsymbol{x}=\boldsymbol{c}$ である。関連グラフ G_{L1} と G_{L2} に対応する局所系はそれぞれ $L_1\boldsymbol{x}'_1=\boldsymbol{c}'_1$ と $L_2\boldsymbol{x}'_2=\boldsymbol{c}'_2$ となる；ここで、$L_1=\begin{pmatrix} b_{22} & b_{24} \\ b_{42} & b_{44} \end{pmatrix}$, $L_2=\begin{pmatrix} b_{33} & b_{35} \\ b_{53} & b_{55} \end{pmatrix}$ であり、$\boldsymbol{x}'_1=(x_2,x_4)^{\mathrm{t}}$, $\boldsymbol{x}'_2=(x_3,x_5)^{\mathrm{t}}$, $\boldsymbol{c}'_1=(c_2-b_{21}x_1,\ c_4)^{\mathrm{t}}$, $\boldsymbol{c}'_2=(c_3-b_{32}x_2,\ c_4-b_{54}x_4)^{\mathrm{t}}$ である。B' が特徴系になることは行列 B からは簡単には分らず、グラフ理論によるアプローチの有効性を示している。 □

5 頂点数100のグラフに対する実験

我々は、前章に述べた局所ブロック化の算法を三つの異なるタイプの例 (下記の 図5, 図6, 図7 を参照) で実験した。各図において、"数 i を含む○" は大域的頂点 v_i を表し、"数 $\mathbf{i}$ を含む四角形" は内部に10個の頂点を含む部分系 (body) を表す。図7において • は非零項 $b_{ji}x_i$ $(i\neq j)$ を表す。部分系の詳細は各例で記述する。

例5と例6では、大域的頂点 $v_1,v_2,\ldots,v_{10}$ は9個の部分系を繋ぐ役割を果たしている。繋がり方は、それぞれの例が特有の性質を持つようにマニュアルで設定した。各部分系は次のようにコンピュータで生成した：第 i 部分系では、変数 $x_{10i+1},x_{10i+2},\ldots,x_{10i+10}$ に関する10個の方程式を、各方程式が (定数項を除いて) 多くの場合3個の、稀には2個の非零項を持ち、第 j 方程式は対角項 $c_{j,10i+j}x_{10i+j}$ を持つようにランダムに生成した。次に、各変数 x_{10i+j} $(1\leq j\leq 10)$ がそれぞれ2個以上の方程式に含まれるか否かをチェックし、含まれない場合には項数の少ない方程式を見つけて項 $c_{j',10i+j}x_{10i+j}$ を追加する (j' は追加される方程式番号)。最後に、得られた部分系が唯一のSCCとなっているかチェックし、もしも複数個のSCCを持つならば棄却する。このチェックで棄却される率は30 %ほどであった。なお、大域的頂点に繋がる辺も、系全体が唯一のSCCを持つように定めることは言うまでもない。

例5 理想的にモジュラー化された系の例 (下図参照)。

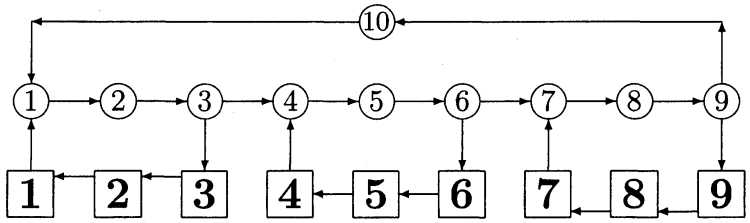

図5 ： 例5における系の全体図 (理想的にモジュラー化)

この例では $S_{\mathrm{char}}=\{\mathrm{Eq}_1,\mathrm{Eq}_2,\ldots,\mathrm{Eq}_{10}\}$ となるべきだろう。そこで、$U=(10,1,9,10)$ とすれば *findSpaths* は次の経路集合を返す: (((10,1)), ((1,2,3,4,5)), ((5,6,7,8,9)), ((9 10)). これより特徴系を表す周回路として次が得られる: $G_{\mathrm{cycl}}=(10,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10)$. 次に、無条件で *setAdjrem* を起動すると、Adj から $(v_i\to v_{i+1})$ $(i\leq 9)$ が削除され、

Adjrem[i] = ( ) ($i \in \{1,2,4,5,7,8\}$), Adjrem[3] = (31), Adjrem[6] = (61), Adjrem[9] = (91) となる。最後に Adjrem に *SCCdecomp* を適用すると下記の SCC が得られた (簡単のため、各 SCC は頂点の集合として表す)。

$$
\begin{aligned}
&\mathrm{SCC}_{\mathrm{single}} = \{1\},\ \{2\},\ \ldots,\ \{10\},\\
&\mathrm{SCC}_{11} = \{11,\ 14,\ 12,\ 18,\ 15,\ 20,\ 16,\ 17,\ 13,\ 19\},\\
&\mathrm{SCC}_{21} = \{21,\ 24,\ 22,\ 25,\ 26,\ 28,\ 23,\ 27,\ 29,\ 30\},\\
&\quad\ldots \qquad\qquad \ldots \qquad\qquad \ldots\\
&\mathrm{SCC}_{81} = \{90,\ 81,\ 87,\ 84,\ 83,\ 89,\ 85,\ 82,\ 86,\ 88\},\\
&\mathrm{SCC}_{91} = \{91,\ 92,\ 96,\ 95,\ 100,\ 93,\ 97,\ 94,\ 98,\ 99\}.
\end{aligned}
$$

望みどおり、すべての部分系が完全に分離された。

例6 現実的な系に近い (???) 例 (下図参照)

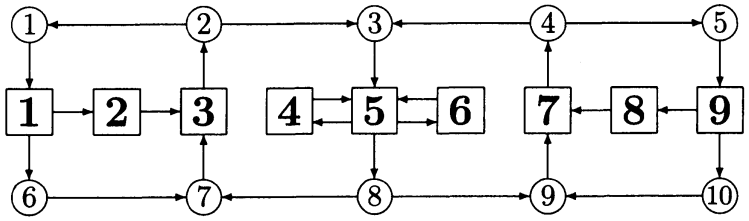

図6 : 例6における系の全体図 (より現実的?)

図は縦線 ③→[5]→⑧ に関して対称なので、$U := (1,\ 6,\ 7,\ 2,\ 1)$ として左側の3部分系の分離を試みる。すると、*findSpaths* は paths = (((1,11,20,6)), ((6,7)), ((7,40,33,31,2)), ((2,1))) を返すので、$G_{\mathrm{cycl}} = (1,\ 11,\ 20,\ 6,\ 7,\ 40,\ 33,\ 31,\ 2,\ 1)$ が得られる。*setAdjrem* により無条件で G_{rem} を計算し、G_{rem} を *SCCdecomp* で SCC 分解すると、次が得られた。

$$
\begin{aligned}
&\mathrm{SCC}_{\mathrm{single}} = \{1\},\ \{7\},\ \{2\},\ \{6\},\\
&\mathrm{SCC}_{11} = \{11,\ 14,\ 12,\ 18,\ 15\},\\
&\mathrm{SCC}_{19} = \{13,\ 20,\ 16,\ 17,\ 19\},\\
&\mathrm{SCC}_{21} = \{25,\ 26,\ 22,\ 28,\ 23,\ 21,\ 24,\ 29,\ 30,\ 27\},\\
&\mathrm{SCC}_{31} = \{35,\ 32,\ 34,\ 40,\ 38,\ 31,\ 36,\ 33,\ 39,\ 37\},\\
&\mathrm{SCC}_{456789} = \{\text{composed of other 66 vertices}\}.
\end{aligned}
$$

部分系 1 が二つの SCCs に分割されるのは望ましくない。

上で望ましくない結果が得られたのは G_{rem} を無条件で計算したためである。そこで、

Cond=(1,10) として計算し、*SCCdecomp* にかけると次の結果が得られた。

$$\begin{aligned}
&\mathrm{SCC}_{\mathrm{single}} = \{2\},\ \{6\},\ \{1\},\ \{7\},\\
&\mathrm{SCC}_1 = \{11,\ 14,\ 12,\ 18,\ 15,\ 20,\ 16,\ 17,\ 13,\ 19\},\\
&\mathrm{SCC}_2 = \{25,\ 26,\ 22,\ 28,\ 23,\ 21,\ 24,\ 29,\ 30,\ 27\},\\
&\mathrm{SCC}_3 = \{35,\ 32,\ 34,\ 40,\ 38,\ 31,\ 36,\ 33,\ 39,\ 37\},\\
&\mathrm{SCC}_{456789} = \{\text{composed of other 66 vertices}\}.
\end{aligned}$$

今度の結果は望ましいものである。なお、SCC_{456789} は大きいが、算法を再度適用すれば部分系に分割できる。($U = (2,\ 6,\ 3,\ 8,\ 4,\ 10,\ 4)$ として G_{cycl} を計算してもよいが、現在のプログラムは *setAdjrem* に複数の条件を入れられないので、実験していない)。

例7 部分系をランダムに結合した複雑な例 (下図参照)

本例は上の2例とは非常に異なる。第 i 部分系は、変数 $v_{10i-9}, v_{10i-8}, \ldots, v_{10i}$ からなることを除き、その作成法は同じである。違いは各部分系が部分系外の20個の非対角項 $b_{10i-i',10j-j'}x_{10j-j'}$ $(0 \le i', j' \le 9)$ でランダムかつ緩く結合していることである。非対角項 (辺に対応し、図では • で表されている) は次のように生成した。各部分系では系内の一つの方程式が非対角項を二つづつ含み、添字 i' と j' を次のように定めた：$j \le 5$ のときは $i' = 0,\ j' = 9$ とし (Adj[$10i$] に辺 $(v_{10i} \to v_{10j-9})$ を追加する)、$j \ge 6$ のときは $i' = 9,\ j' = 0$ とする (Adj[$10i - 9$] に辺 $(v_{10i-9} \to v_{10j})$ を追加する)。参考のため、Adj の一部を示す：$\mathrm{Adj}[10] = (91, 61, \ldots)$, $\mathrm{Adj}[20] = (71, 21, \ldots)$, $\ldots$, $\mathrm{Adj}[81] = (20, 80, \ldots)$, $\mathrm{Adj}[91] = (30, 50, 60, \ldots)$.

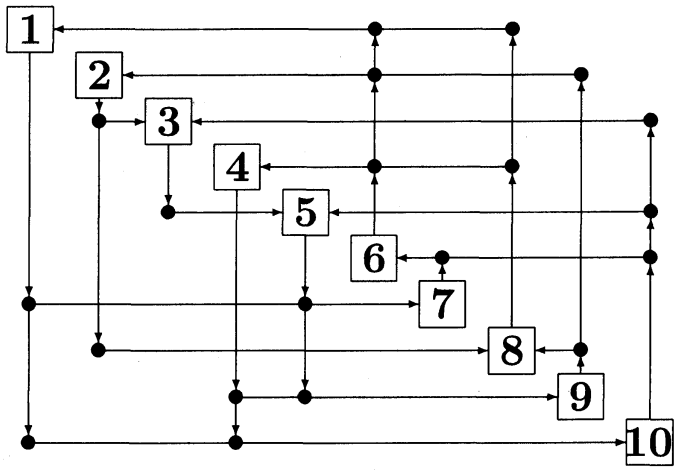

図7： 例7における系の全体図 (ややランダムに生成)

図7から特徴系を見つけるのは困難なので、試みに $U := (1,\ 100,\ 1)$ として探してみる。すると、*findSpaths* は次の経路集合を返す：Spaths = (((1, 10, 91, 98, 93, 9, 100)), ((100, 92, 91, 60, 52, 51, 10, 9, 2, 1))). これより、特徴系を表す周回路として $G_{\mathrm{cycl}} =$

(1, 10, 91, 98, 93, 99, 100, 92, 91, 60, 52, 51, 10, 9, 2, 1) が得られた。これは第1部分系と第10部分系の内部を複雑に辿るので良くない。辿る理由は頂点v_1とv_{100}を通過するように指定したからである;実際、頂点v_{10}とv_{91}を2回づつ通過する。そこで、$U = (10, 91, 10)$と指定して再計算した。今度は$G_{\mathrm{cycl}} = (10, 91, 60, 52, 51, 10)$が得られた。条件なしで$G_{\mathrm{rem}}$を計算し、SCC分解すると、頂点$v_{58}$と$v_{60}$以外に次の成分が得られた。

$$\mathrm{SCC}_1 = \{1, 4, 2, 8, 5, 10, 6, 7, 3, 9\},$$
$$\mathrm{SCC}_5 = \{51, 53, 59, 55, 52, 54, 56, 57\},$$
$$\mathrm{SCC}_6 = \{61, 62, 68, 69, 63, 65, 67, 66, 70, 64\},$$
$$\mathrm{SCC}_{2347890} = \{\text{composed of other vertices}\}.$$

$\mathrm{SCC}_{2347890}$は算法を再帰的に適用して更に分割することができる。しかし、頂点v_{58}とv_{60}を第5部分系に含めようと上記のG_{rem}を Cond='(51,60) なる条件で *setAdjrem* にかけてみたが、含めることはできなかった。

6 終わりに

本稿のテーマに取り掛かった当初は、パラメータ係数の連立線形方程式の解について、包括的グレブナー基底と同じような理論を作れば終わりだと単純に考えていた。その件については、第2章で述べた「パラメータ空間での解の縮退」で話は済んだも同然である。係数行列のごく一部を取り出して方程式系を局所的に解くことは1963年にGabriel[4]が提唱しており、局所ブロックはそれをグラフ理論で定式化したものである。局所ブロックを強連結グラフで定義した頃から研究は予定しない方向に進んだ。

本稿の“売り”は、特徴系を応用に即して定義し算法化したことと、よく知られたSCC算法を用いた局所ブロック化法を考案したことである。論文[8]でレフェリーの一人から「私はこの論文のアイデアは好きだが、算法は嫌いだ。なぜなら…」と言われ、短期間で面子をかけて考案したのが[7]のアイデアと算法である。アイデアが的を得ているか気になるし、算法も産業界の実課題でテストしたわけではないが、グラフ理論による定式化と関連算法の開発は終了した。今後は、パラメータ入り微分代数方程式系の数式処理的解法の開発に取り組む予定だが、机上の空論に陥らないことが肝要だろう。

前途には課題が山積している。微分代数方程式は数値計算には非常に悪条件であることが昔から知られており、数値解析の伝統的アプローチから取り組んだ研究者らは四苦八苦していた。数値解析のこの難題は、代数方程式を数式のままで微分して簡単化するという数式処理的アプローチで見事に解決された[1]。しかし、この解決法も「精度よく解ける常微分方程式系に変換して数値解法で解く」というもので、数式処理を使ってはいるもののフルに活用したというわけではない。また、産業数学分野では、実課題に精通しているだけに、非常に巧妙で微細な算法がいくつも開発されている[1]。しかし、計算機代数の最新の成果を利用しているわけではない。なんとか、計算機代数の成果をフルに活用するような算法を開発したいものである。

参 考 文 献

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

[2] I.S. Duff and J.K. Reid: An implementation of Tarjan's algorithm for the block triangularization of a matrix. ACM Tarns. Math. Soft. 4 (1978), 137-147.

[3] A.C. Dulmage and N.S. Mendelsohn: Coverings of bipartite graph. Canad. J. Math. 10 (1958), 517-534; A structure theorem of bipartite graphs of finite exterior dimension. Trans. Roy. Soc. Canad. Sec. III 53 (1959), 1-13.

[4] K. Gabriel: *Diakoptics: The Piecewise Solution of Large-Scale Systems.* Macdonald Publishing, Kondon, 1963.

[5] K. Murota: *Matrices and Matroids for Systems Analysis.* Springer-Verlag, Berlin, 2000.

[6] A. Pothen and C-J. Fan: Computing the block triangular form of a sparse matrix. ACM Tarns. Math. Soft. 16 (1990), 303-324.

[7] T. Sasaki: Solving parametric sparse linear systems by local blocking, II. Proceedings of SYNASC2014 (Timisoara, Romania); SYNASC2014 (2015), 74-81, IEEE.

[8] T. Sasaki, D. Inaba and F. Kako: Solving parametric sparse linear systems by local blocking. Proceedings of CASC2014 (Warsaw, Poland); LNCS 8660 (2014), Springer.

[9] T. Sasaki and T. Yamaguchi: On algebraic preprocessing of floating-point DAEs for numerical model simulation, Proceedings of SYNASC2013 (Timisoara, Romania); SYNASC 2013 (2014), 81-88, IEEE.

[10] 佐々木建昭, 山口 哲: パラメータ入りの連立線形方程式の誤差低減法. 数理解析研究所講究録 1907 号, 2014 年 7 月, 20-31.

[11] R.E. Tarjan: Depth-first search and linear graph algorithms. SIAM J. Computing. 1 (1972), 146-160.

[12] D.B. West: Introduction to Graph Theory (2nd ed.), Chap. 3, Prentice Hall, 1999.