数理解析研究所講究録 第2094巻 2018年 122-130

連立線型方程式に対する定常反復解法の並列計算に関する一考察

京都大学大学院 情報学研究科　藤原宏志
Hiroshi Fujiwara
Graduate School of Informatics,
Kyoto University

1　緒言

本研究では連立一次方程式の反復解法に対し，その並列計算で生じ得る競合状態 (race condition) 下での計算の信頼性を目的とした計算過程について論じる．

偏微分方程式や積分方程式の離散化では大規模な連立一次方程式が現れるが，ハードウエアの進歩にともなってその反復解法の並列計算が一般的となっている．特に近年は，CPU に加えてグラフィックス用プロセサも含めたヘテロジニアスな計算環境ももちいられ，多層化するメモリへのアクセスも込めて複雑化する計算環境の性能を最大限に活かすことが求められている．

高速・高効率な並列計算のためには実行中の計算機の状態に応じた動的な自動負荷分散が有効と考えられる．しかし適切に制御されていない並列化では，計算結果が実行状態に依存して計算のたびに変わり得る競合状態が生じ，数値計算の信頼性が損われる．一方，定常反復解法の多くは逐次計算を念頭においた議論が多いが，それらの理論，特に収束性が並列計算でも再現されるとは限らない．そのため，逐次計算では収束性がある設定でも，その並列計算で理論に沿う計算を実現するためには，アルゴリズム (漸化式) のフロー依存の解析により結果の変化を招く並列化を禁止したり，競合状態を生じないように排他制御や同期が必要となる場合があり，計算速度の低下を招き得る．しかしながら科学・技術数値計算として本来要求されるのは，結果の一貫性よりも，高速かつ (理論に基く) 信頼性の高い計算結果である．

物理や工学現象に端を発する連立一次方程式を例にすると，その係数行列が狭義優対角や正定値対称であることが多く，この場合，例えば Gauss-Seidel 法が収束する．これはメモリを効率的に利用するスキームで大規模問題に適することはよく知られているものの，計算順序が厳密に決められており，逐次計算で収束する問題設定であってもその単純な並列化が収束するとは限らない．従来は Gaussi-Seidel 法の並列化のためにブロック Gauss-Seidel 法，加速法を併用した非同期逐次緩和 (asynchronous SOR) や，計算順の変更の検出と補正法等が研究されているが，従来の逐次計算の理論と異なり，収束のための決定的な指針は著者の知る限り与えられていない．

そこで本研究では，競合状態を防ぐのではなく，動的な負荷分散での競合状態下の計算の信頼性を論ずるための数理モデルを示す．ただし簡単のため，データ競合は生じないと仮定する[1]．

[1] 並列処理において，共有リソースへのアクセスを含む処理が複数のタスク (スレッドやプロセス) で同時に実行されると，結果はそれらの実行順序に依存して実行のたびに異なり，非決定的となる．これを競合状態 (race condition) という [1]．例えば複数スレッドからの共有変数への書き込み処理や，ひとつのスレッドからのみの参照と変更が想定されるようなカウンタなどの状態を表す変数に依存する (さらにその変数の値を変更する) 処理が，適切に制御されず同時に実行されてしまう場合が考えられる．

2 Gauss-Seidel 法の共有メモリ並列計算の実行過程

2.1 Gauss-Seidel 法とその実装例

例として連立一次方程式

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

を考える．ただし対角成分 a_{ii} はすべて 0 でなく，かつ解 (x_*, y_*, z_*) がただひとつ存在すると仮定する．このとき Gauss-Seidel 法は，初期値 (x_0, y_0, z_0) を与えて

$$x_{k+1} = (b_1 - a_{12}\, y_k - a_{13}\, z_k)/a_{11}, \tag{2.1a}$$

$$y_{k+1} = (b_2 - a_{21}\, x_{k+1} - a_{23}\, z_k)/a_{22}, \tag{2.1b}$$

$$z_{k+1} = (b_3 - a_{31}\, x_{k+1} - a_{32}\, y_{k+1})/a_{33} \tag{2.1c}$$

で列 $\{(x_k, y_k, z_k)\}$ を生成するもので，適当な条件のもとで $k \to \infty$ のとき (x_*, y_*, z_*) に収束する．以下，本論文を通じて，添字 k は反復の回数を表すものとする．

Gauss-Sidel 法 (2.1) は (2.1a), (2.1b), (2.1c) の順で実行するが，(x_k, y_k, z_k) と $(x_{k+1}, y_{k+1}, z_{k+1})$ の出現を考えるとこれらを同時に保持する必要はなく，後者の各成分が求まるたびに前者の対応するメモリ領域 (変数) に上書きしてよい．したがってメモリを節約することができ，Jacobi 法などに比べて大規模計算において大きな利点となる．b の値を a に代入することを記号 $a \leftarrow b$ で表すことにすると，(2.1) の実装としては次が自然である．

$$x \leftarrow (b_1 - a_{12}\, y - a_{13}\, z)/a_{11}, \tag{2.2a}$$

$$y \leftarrow (b_2 - a_{21}\, x - a_{23}\, z)/a_{22}, \tag{2.2b}$$

$$z \leftarrow (b_3 - a_{31}\, x - a_{32}\, y)/a_{33}. \tag{2.2c}$$

ただし (2.2a), (2.2b), (2.2c) はフロー依存をもつため，この順で実行しなければならない．

2.2 共有メモリ並列計算の実行過程

プログラム (2.2) を，メモリを共有する 2 つのスレッド $T1, T2$ で計算することを考える．ただし簡単のため，以下の条件での処理を考える．

- (2.2a), (2.2b) はそれぞれ $T1, T2$ で実行される．
- (2.2c) も $T1$ と $T2$ のいずれか一方のみで実行される．
- 反復ごとに $T1, T2$ は同期する，すなわち (2.2c) の処理が終了してから次の反復の (2.2a) の処理を始める．

これらの条件のもとでプログラム (2.2) が並列に実行される過程を調べる．以下，$T1 > T2$ などの不等号は，メモリアクセスも含めたスレッドの処理速度の比較とする．

一方，複数のスレッドから，ある共有データに対するメモリアクセスが同時に発生し，かつ少なくともひとつが書き込みであるとき，データ競合 (data race) が生じ得る [2]．例えば，あるスレッドがある共有変数を更新中に，他のスレッドがその変数を読み出す場合が考えられる．データ競合の結果の状態は未定義と定められることが多い．

並列計算過程 **1** ($T1 = T2$). $T1, T2$ が同じ速度ならば, (2.2a), (2.2b) は次の過程で処理される.

(1) $T1$ が $y(= y_k)$ を読み込む.

(2) $T2$ が $x(= x_k)$ を読み込む.

(3) $T1$ が $z(= z_k)$ を読み込む.

(4) $T2$ が $z(= z_k)$ を読み込む.

(5) $T1$ が (2.2a) の右辺を計算し, 結果 $(= x_{k+1})$ を x に書き込む.

(6) $T2$ が (2.2b) の右辺を計算し, 結果 $(= y_{k+1})$ を y に書き込む.

残る (2.2c) が $T1$ で計算されるならば, 続けて次が実行される.

(7) $T1$ が $x(= x_{k+1})$ を読み込む.

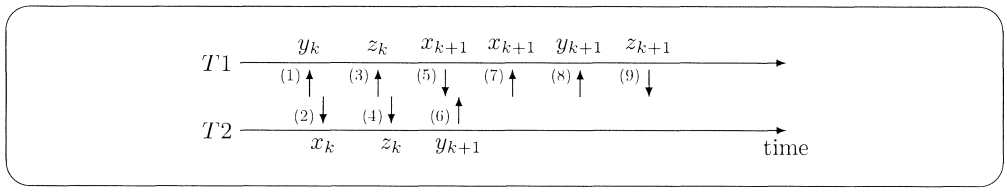
(8) $T1$ が $y(= y_{k+1})$ を読み込む.

(9) $T1$ が (2.2c) の右辺を計算し, 結果 $(= z_{k+1})$ を z に書き込む.

このうち (2) は, Gauss-Seidel 法では反復で更新された x_{k+1} をもちいる. 一方ここでは, x_{k+1} を求める $T1$ における処理 (5) に先立って (2) が実行されることを想定しており, (2) では更新前の x_k をもちいることになる. したがってこの過程は Gauss-Seidel 法と異なるものであり, (2.2c) が同様のタイミングで $T2$ で実行される場合も含めて次の漸化式で表される.

$$
\begin{aligned}
T1 \quad &: \quad x_{k+1} = (b_1 - a_{12}\, y_k - a_{13}\, z_k)/a_{11}, \\
T2 \quad &: \quad y_{k+1} = (b_2 - a_{21}\, x_k - a_{23}\, z_k)/a_{22}, \\
T1 \text{ or } T2 \quad &: \quad z_{k+1} = (b_3 - a_{31}\, x_{k+1} - a_{33}\, y_{k+1})/a_{33}.
\end{aligned}
$$

この過程を次の図で表す. ただし横軸に計算時刻の経過を, ↑, ↓ は各スレッドのメモリアクセス (読み込みもしくは書き込み) を表すものとする.

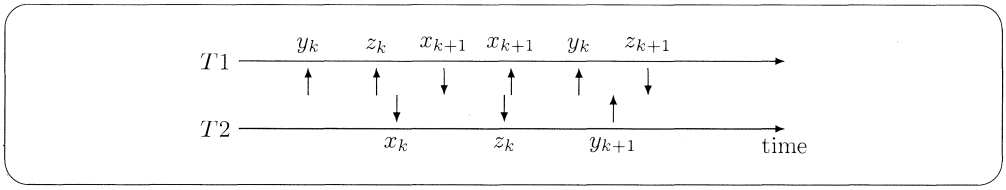


さて並列計算過程を表す漸化式の右辺における x_k, x_{k+1} などの出現を考えると, 上述の条件のものとで起り得るのは過程 1 に加えて以下の 8 つの場合である.

並列計算過程 **2** ($T1 > T2$).

$$
\begin{aligned}
T1 \quad &: \quad x_{k+1} = (b_1 - a_{12}\, y_k - a_{13}\, z_k)/a_{11}, \\
T2 \quad &: \quad y_{k+1} = (b_2 - a_{21}\, x_k - a_{23}\, z_k)/a_{22}, \\
T1 \quad &: \quad z_{k+1} = (b_3 - a_{31}\, x_{k+1} - a_{33}\, y_k)/a_{33}.
\end{aligned}
$$

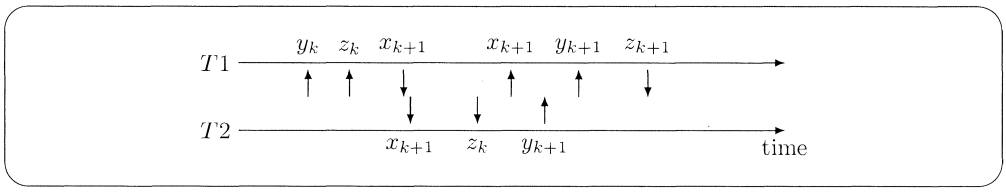
これは, 過程 1 の場合に比べて $T1$ が $T2$ より高速に処理された場合であり, 次の図で表されるものが考えられる.



並列計算過程 **3** ($T1 > T2$).

$$\begin{aligned} T1 &: x_{k+1} = (b_1 - a_{12}\, y_k - a_{13}\, z_k)/a_{11}, \\ T2 &: y_{k+1} = (b_2 - a_{21}\, x_{k+1} - a_{23}\, z_k)/a_{22}, \\ T1 &: z_{k+1} = (b_3 - a_{31}\, x_{k+1} - a_{33}\, y_{k+1})/a_{33}. \end{aligned}$$

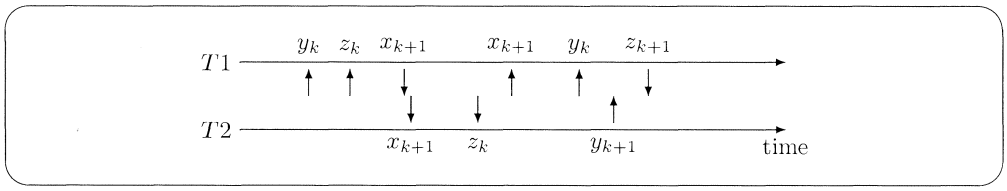
これは $T2$ における x の読み込みが遅れた場合などで，例えば次の図が実現例である.



並列計算過程 **4** ($T1 \gg T2$).

$$\begin{aligned} T1 &: x_{k+1} = (b_1 - a_{12}\, y_k - a_{13}\, z_k)/a_{11}, \\ T2 &: y_{k+1} = (b_2 - a_{21}\, x_{k+1} - a_{23}\, z_k)/a_{22}, \\ T1 &: z_{k+1} = (b_3 - a_{31}\, x_{k+1} - a_{33}\, y_k)/a_{33}. \end{aligned}$$

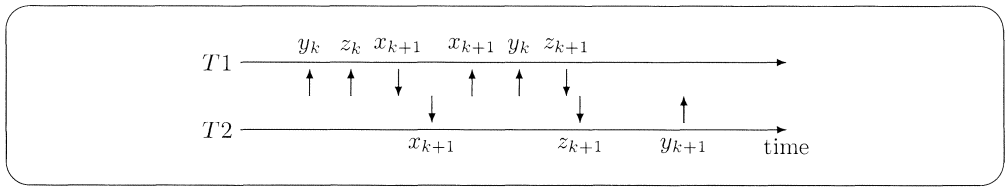
これは過程 3 で，$T2$ における (2.2b) の y_{k+1} の書き込みよりも $T1$ での y_k の読み込みが速い場合で，例えば次の実行を表す.



並列計算過程 **5** ($T1 \ggg T2$).

$$\begin{aligned} T1 &: x_{k+1} = (b_1 - a_{12}\, y_k - a_{13}\, z_k)/a_{11}, \\ T2 &: y_{k+1} = (b_2 - a_{21}\, x_{k+1} - a_{23}\, z_{k+1})/a_{22}, \\ T1 &: z_{k+1} = (b_3 - a_{31}\, x_{k+1} - a_{33}\, y_k)/a_{33}. \end{aligned}$$

これは，過程 4 よりもさらに $T1$ が高速な場合であり，次の時系列が対応する.



また $T1$ と $T2$ の，メモリアクセスも含めた処理速度が逆になった場合，上の過程 2 から 5 に対応して，次の 4 つの過程が考えられる．

並列計算過程 **6** ($T1 < T2$).

$$\begin{aligned} T1 &: x_{k+1} = (b_1 - a_{12}\, y_k - a_{13}\, z_k)/a_{11}, \\ T2 &: y_{k+1} = (b_2 - a_{21}\, x_k - a_{23}\, z_k)/a_{22}, \\ T2 &: z_{k+1} = (b_2 - a_{31}\, x_k - a_{33}\, y_{k+1})/a_{33}. \end{aligned}$$

並列計算過程 **7** ($T1 < T2$).

$$\begin{aligned} T1 &: x_{k+1} = (b_1 - a_{12}\, y_{k+1} - a_{13}\, z_k)/a_{11}, \\ T2 &: y_{k+1} = (b_2 - a_{21}\, x_k - a_{23}\, z_k)/a_{22}, \\ T2 &: z_{k+1} = (b_2 - a_{31}\, x_{k+1} - a_{33}\, y_{k+1})/a_{33}. \end{aligned}$$

並列計算過程 **8** ($T1 \ll T2$).

$$\begin{aligned} T1 &: x_{k+1} = (b_1 - a_{12}\, y_{k+1} - a_{13}\, z_k)/a_{11}, \\ T2 &: y_{k+1} = (b_2 - a_{21}\, x_k - a_{23}\, z_k)/a_{22}, \\ T2 &: z_{k+1} = (b_2 - a_{31}\, x_k - a_{33}\, y_{k+1})/a_{33}. \end{aligned}$$

並列計算過程 **9** ($T1 \lll T2$).

$$\begin{aligned} T1 &: x_{k+1} = (b_1 - a_{12}\, y_{k+1} - a_{13}\, z_{k+1})/a_{11}, \\ T2 &: y_{k+1} = (b_2 - a_{21}\, x_k - a_{23}\, z_k)/a_{22}, \\ T2 &: z_{k+1} = (b_2 - a_{31}\, x_k - a_{33}\, y_{k+1})/a_{33}. \end{aligned}$$

注意．形式的には上記以外に，例えば

$$\begin{aligned} x_{k+1} &= (b_1 - a_{12}\, y_{k+1} - a_{13}\, z_k)/a_{11}, \\ y_{k+1} &= (b_2 - a_{21}\, x_k - a_{23}\, z_k)/a_{22}, \\ z_{k+1} &= (b_2 - a_{31}\, x_k - a_{33}\, y_k)/a_{33} \end{aligned}$$

という漸化式も考えられるが，上述の条件のもとではこれに対応する過程は生じない．

Gauss-Seidel 法の逐次計算を想定した実装 (2.2) に対する並列計算の過程としては以上で示す 9 個が考えられる．このうち Gauss-Seidel 法に一致するのは過程 3 のみである．また，Gauss-Seidel 法が収束する場合でも他の計算過程が収束性をもつとは限らない．

例 **1** (Gauss-Seidel 法が収束し，並列計算が収束しない例)．$A = \begin{pmatrix} 3 & -2 & -2 \\ -2 & 2 & 1 \\ -2 & 1 & 2 \end{pmatrix}$ とし，連立方程式 $Ax = b$ を考える．A の固有値は $1, 3 \pm \sqrt{8}$ のため A は正定値対称であり，$Ax = b$ に対する Gauss-Seidel 法は任意の初期値に対して解に収束する．一方，過程 6 の反復計算は次で表される．

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \\ z_{k+1} \end{pmatrix} = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 2 \end{pmatrix}^{-1} \left\{ \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} + \begin{pmatrix} 0 & 2 & 2 \\ 2 & 0 & -1 \\ 2 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix} \right\}.$$

この反復行列 $\begin{pmatrix}3&0&0\\0&2&0\\0&1&2\end{pmatrix}^{-1}\begin{pmatrix}0&2&2\\2&0&-1\\2&0&0\end{pmatrix}=\begin{pmatrix}0&\frac{2}{3}&\frac{2}{3}\\1&0&-\frac{1}{2}\\\frac{1}{2}&0&\frac{1}{4}\end{pmatrix}$ の固有値の近似値は -1.031990, $0.937430, 0.344560$ であり，スペクトル半径は 1 を越える．したがってこの反復は収束しない [3].

3 定常反復解法の並列計算の数理モデル

前節では共有メモリ並列計算の反復解法の 1 回の反復の過程を考えた．しかし一般には，スレッドの処理速度は反復ごとに同一とは限らず，動的な負荷分散が起こると反復ごとに異なる過程をとる可能性がある．さらに実際の数値計算では共有メモリ型並列計算のみならず，分散メモリ型並列計算やそれらを同時にもちいるハイブリッド並列ももちいられ，計算過程は極めて複雑となる．本節では，それらの解析のための並列計算過程を表す数理モデルを示す．

$A=\left(a_{ij}\right)$, $a_{ij}\in\mathbb{C}$ を対角要素がすべて 0 でない N 次正則行列とし，

$$A=D-P-S$$

を考える．ただし $D=\mathrm{diag}(a_{11},\ldots,a_{NN})$ は A の対角成分からなる対角行列で，$P=\left(p_{ij}\right)$, $S=\left(s_{ij}\right)$ はすべての i で $p_{ii}=s_{ii}=0$ であって，$a_{ij}=0\,(i\neq j)$ のときは $p_{ij}=s_{ij}=0$, $a_{ij}\neq 0\,(i\neq j)$ のときは次のいずれか一方を満たすものとする．

$$\begin{cases}p_{ij}=-a_{ij},\\ s_{ij}=0,\end{cases}\quad\text{または}\quad\begin{cases}p_{ij}=0,\\ s_{ij}=-a_{ij}.\end{cases}$$

この $A=D-P-S$ を A の加法的分解という．さらに添字の集合を $P_i=\{j\,;\,j\neq i, p_{ij}\neq 0\}$, $S_i=\{j\,;\,j\neq i, s_{ij}\neq 0\}$ で定めると，$P_i\cap S_i=\emptyset$ かつ $P_i\cup S_i=\{j\,;\,1\leq j\leq N,\, j\neq i,\, a_{ij}\neq 0\}$ である．

さて $Ax=b$ に対する Gauss-Seidel 法を k 回反復して得られる $\mathbb{C}^N$ の元を $x^{(k)}=\left(x_1^{(k)},\ldots,x_N^{(k)}\right)$ とする．本研究の主張は，並列計算の過程を表す次の数理モデルである．

定理 2. *Gauss-Seidel* 法の実装に対し，データ競合が生じず，反復ごとに同期すると仮定する．この並列計算過程は次で表される: 反復 k ごとに添字の集合 $P_1^{(k)},\ldots,P_N^{(k)}$ (したがって $S_1^{(k)},\ldots,S^{(k)}$) がただひとつ存在して，

$$x_i^{(k+1)}=\frac{1}{a_{ii}}\left(b_i-\sum_{j\in S_i^{(k)}}a_{ij}x_j^{(k+1)}-\sum_{j\in P_i^{(k)}}a_{ij}x_j^{(k)}\right).$$

すなわち，並列計算過程に列 $\left\{P^{(1)},P^{(2)},\ldots\right\}$ (したがって $\left\{S^{(1)},S^{(2)},\ldots\right\}$) が対応する．またこれは反復 k ごとに加法的分解 $A=D-P^{(k)}-S^{(k)}$ がただひとつ定まり，

$$x^{(k+1)}=\left(D-S^{(k)}\right)^{-1}\left(b+P^{(k)}x^{(k)}\right).\tag{3.1}$$

と表されることと同値である．

注意．このモデルは形式的に $P_i=\emptyset$ も含まれるが，$A=D$ の自明な場合を除き，これに対応する過程は実際には生じない．

例 **3.** A の対角成分を含まない下三角成分，上三角成分をそれぞれ L, U とする．簡単のため $a_{ij} \neq 0$ とする．

(1) Jacobi 法は任意の k に対して $S_i^{(k)} = \emptyset$, $P_i^{(k)} = \{1, 2, \ldots, N\} \setminus \{i\}$, すなわち $S^{(k)}$ は零行列, $P^{(k)} = L + U$ である．

(2) Gauss-Seidel 法は任意の k に対して $S_i^{(k)} = \{j\,;\, j < i\}$, $P_i^{(k)} = \{j\,;\, j > i\}$, すなわち $S^{(k)} = L$, $P^{(k)} = U$ である．

4 収束の充分条件

本節では並列計算の信頼性のために，(3.1) で列 $\{x^{(k)}\}$ が定まり，かつ収束する充分条件を論じる．

まず正方行列 A に対して次を考える．

$$\mathcal{M}_A = \{(D-S)^{-1}P\,;\, A = D - P - S \text{ は加法的分解で，} D - S \text{ が正則なもの}\}.$$

ただし計算過程が存在すれば $D - S$ は正則であるが，逆は成立しないなど，$\mathcal{M}_A$ のすべての元に計算過程が対応するわけではないことに注意する．

さてこの $\mathcal{M}_A$ は有限集合であり，$\lambda^* = \max_{M \in \mathcal{M}_A} \rho(M)$ が存在する．ただし $\rho(M)$ は M のスペクトル半径とする．また任意の正方行列 M と正数 ϵ に対し，subordinate な行列ノルム $\|\cdot\|$ が存在して $\|M\| \leq \rho(M) + \epsilon$ とできる [4].

補題 **4.** A が狭義優対角とする．任意の $M \in \mathcal{M}_A$ に対して $\rho(M) < 1$ である．

Proof. $(D-S)^{-1}P$ の固有値のひとつを λ, それに属する固有ベクトルを $v \neq 0$ とし，$|v_1|, \ldots, |v_N|$ の最大値を $|v_i|$ とする．$(D-S)^{-1}Pv = \lambda v$, すなわち $Pv = \lambda(D-S)v$ であり，その i 行は

$$\sum_{j \in P_i} -a_{ij}v_j = \lambda \left\{ a_{ii}v_i + \sum_{j \in S_i} a_{ij}v_j \right\}.$$

ここで $|\lambda| \geq 1$ と仮定すると，$|v_j| \leq |v_i|$, $v_i \neq 0$ より

$$|a_{ii}| \leq \frac{1}{|\lambda|} \sum_{j \in P_i} |a_{ij}| + \sum_{j \in S_i} |a_{ij}| \leq \sum_{j \neq i} |a_{ij}|.$$

これは A の狭義優対角性に反する．したがって $|\lambda| < 1$ が従う． □

これより $\lambda^* + \epsilon < 1$ となる正数 ϵ が存在する．このとき任意の $M \in \mathcal{M}_A$ に対して適当な subordinate norm $\|\cdot\|$ によって $\|M\| \leq \lambda^* + \epsilon < 1$ とでき，(3.1) の収束の充分条件として次が得られる．

定理 **5.** A が狭義優対角とする．計算過程 $\{(P^{(k)}, S^{(k)})\}$ が k に依存しないならば，(3.1) で得られる $\mathbb{C}^N$ の列 $\{x^{(k)}\}$ は，任意の初期値に対して解に収束する．

Proof. $P = P^{(k)}$, $S = S^{(k)}$ として，(3.1) は

$$x^{(k+1)} = (D-S)^{-1}b + (D-S)^{-1}Px^{(k)},$$

と書ける．さらに解 $x^* = A^{-1}b$ も

$$x^* = (D-S)^{-1}b + (D-S)^{-1}Px^*$$

をみたすので，

$$x^{(k+1)} - x^* = (D-S)^{-1}P\big(x^{(k)} - x^*\big).$$

$M = (D-S)^{-1}P$ とすると，適当な subordinate norm $\|\cdot\|$ により $\|M\| < 1$ であり，対応する $\mathbb{C}^N$ のノルム $\|\cdot\|$ によって

$$\left\|x^{(k)} - x^*\right\| = \left\|M\big(x^{(k-1)} - x^*\big)\right\|$$

行列ノルムが subordinate であることから

$$\leq \|M\| \left\|x^{(k-1)} - x^*\right\| \leq \|M\|^k \left\|x^{(0)} - x^*\right\|.$$

$\|M\| < 1$ より任意の初期値 $x^{(0)}$ に対して $x^{(k)}$ は x^* に $\|\cdot\|$ で収束する．(有限次元のノルムはすべて同等であることに注意する．) □

一般的な並列計算環境では計算過程を明示的に指定することができ，反復ごと計算過程を同一にすることができる．このように事前に負荷分散を決めて反復ごとに変えなければ，係数行列が狭義優対角な連立一次方程式に対し，Gauss-Seidel 法の実装の並列計算は任意の初期値に対して解に収束する．

5 Concluding Remark

本研究では Gauss-Seidel 法に対する並列計算の解析のため数理モデル (3.1) を示し，それをもちいた収束解析の例を示した．一般にこの反復 (3.1) に対しては

$$x^{(k)} - x^* = M_{k-1}\cdots M_0\big(x^{(k)} - x^*\big), \quad M_i \in \mathcal{M}_A$$

であり，収束性を保証するためには積 $M_{k-1}\cdots M_0$ の性質を調べる必要がある．また，実際の数値計算では，ひとつの式を複数のスレッドやプロセスで計算する場合もあり，それを表す数理モデルについても研究が必要である．

謝辞　本研究遂行にあたり，Craig. C. Douglas 教授 (University of Wyoming, USA) から貴重なご助言を頂きました．また本研究の一部は日本学術振興会科学研究費 (C) 26400198, (A) 16H02155 の助成のもとでおこなわれたものです．

参考文献

[1] M. J. Bach, *The Design of the UNIX Operating System*, Prentice-Hall (1986)

[2] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design, Fifth Edition: The Hardware/Software Interface*, Morgan Kaufmann Publishers Inc. (2013)

[3] R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall (1962)

[4] P. G. Ciarlet, B. Miara and J. -M. Thomas, *Introduction to Numerical Linear Algebra and Optimisation*, Cambridge University Press (1989)

[5] 山本哲朗，行列解析の基礎 — Advanced 線形代数 (SGC ライブラリ 79)，サイエンス社 (2010)