数理解析研究所講究録 1362 巻 2004 年 47-55

拡張 Strassen 法による連立一次方程式の精度保証
Fast Verification for Systems of Linear Equations with the Extended Strassen's Method

早稲田大学 理工学研究科　森山 敦史 (Atsushi Moriyama)
早稲田大学 理工学研究科　荻田 武史 (Takeshi Ogita)
Graduate School of Science and Engineering, Waseda University
日立製作所 エンタープライズサーバ事業部　後 保範 (Yasunori Ushiro)
Enterprise Server Division, Hitachi Ltd.
早稲田大学 理工学部　大石 進一 (Shin'ichi Oishi)
School of Science and Engineering, Waseda University

1 はじめに

本報告では，拡張 Strassen 法 [5] を，$n \times n$ 密行列 A を係数とする連立一次方程式

$$Ax = b \tag{1}$$

に適用した場合に，その数値解 (計算機によって得られた近似解) を精度保証することを考える．密行列系連立一次方程式に対する精度保証法は，大石らの研究 [2] によって既に知られており，区間演算を行列・ベクトル積や行列乗算のレベルで考えることができる．そのため，精度保証の過程でも最適化 BLAS などの高速性がそのまま保持される．

一方，行列乗算を通常より高速に行うことが可能である Strassen 法 [3] はよく知られているが，計算過程が通常よりも複雑になるため，結果として精度保証付き数値計算には不向きであると考えられてきた．しかしながら，Strassen 法を用いることによって行列乗算をさらに高速に精度保証付きで計算する方式が文献 [7] で提案された．

そこで，本報告では

- 拡張 Strassen 法による高速な行列乗算
- 近似逆行列による連立一次方程式の数値解の精度保証

以上の 2 点に注目して拡張 Strassen 法を用いた連立一次方程式の数値解の精度保証方式を提案する．この方式は，拡張 Strassen 法を連立一次方程式の解法であるブロック LU 分解法に適用した場合にも有効である．また，数値実験によって本提案方式の有効性を示す．

2 拡張 Strassen 法による行列乗算

連立一次方程式の精度保証に入る前に，拡張 Strassen 法による行列乗算について簡潔に述べる．密行列乗算 $D = B \cdot C$ に Strassen 法を 1 回適用すると，B, C が実行列の場合，通常の計算方法に比較して演算量が 7/8 まで減少することが知られている．Strassen 法では行列

B, C, D をそれぞれ 2×2 ブロック行列に分割して考えるのに対し，拡張 Strassen 法では，図 1 のように，それぞれ $N\times2$, $2\times N$ 及び $N\times N$ ブロック行列に分割して考える．

$$\begin{pmatrix} D_{11} & D_{12} & \cdots & D_{1N} \\ D_{21} & D_{22} & \cdots & D_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ D_{N1} & D_{N2} & \cdots & D_{NN} \end{pmatrix} = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \\ \vdots & \vdots \\ B_{N1} & B_{N2} \end{pmatrix} \cdot \begin{pmatrix} C_{11} & C_{12} & \cdots & C_{1N} \\ C_{21} & C_{22} & \cdots & C_{2N} \end{pmatrix}$$

図 1: 拡張 Strassen 法による行列乗算のモデル

拡張 Strassen 法による行列乗算のアルゴリズムを以下に示す．ここでは，アルゴリズムを MATLAB のように表現する．

Algorithm 1 (後 [5]) 拡張 *Strassen* 法による行列乗算 $B\cdot C$:

> **function** $D = \mathbf{ExtStrass}(B, C, N)$
> **for** $i = 1 : N-3$
> $s = N - i + 1;$
> $U_i = B_{i1} * (C_{1s} + C_{2s});\quad V_i = (B_{i2} - B_{i1}) * C_{2s};$
> $D_{is} = U_i + V_i;$
> **end**
> **for** $j = N-1, 1, -1$
> $k = N - j + 2 * \mathbf{mod}(j, 2);$
>   **if** $(j = 1)$
> $k = N;$
> $L = (B_{k1} - B_{k2}) * C_{1j};\quad M = B_{k2} * (C_{1j} + C_{2j});$
> $D_{kj} = L + M;$
>   **end**
>   **for** $i = 1 : N - j - \mathbf{mod}(j-1, 2)$
> $s = N - i + 1;\quad t = i;$
>   **if** $(i \geq N-2)$
> $t = 1;$
> $U_1 = B_{i1} * (C_{1s} + C_{2s});\quad V_1 = (B_{i2} - B_{i1}) * C_{2s};$
> $D_{is} = U_1 + V_1;$
>   **end**
> $P = (B_{i1} + B_{k2}) * (C_{1j} + C_{2s});\quad Q = (B_{i2} + B_{k2}) * (C_{2j} - C_{2s});$
> $D_{ij} = P + Q - M + V_t;$
> $Q = (B_{i1} + B_{k1}) * (C_{1s} - C_{1j});$
> $D_{ks} = P + Q + L - U_t;$
>   **end**
> **end**

$N \geq 2$ のとき，拡張 Strassen 法を実行列乗算に1回適用すると，通常の計算方法に比較して，理論的には演算量が 3/4 まで減少する．

3 精度保証理論

本章では，拡張 Strassen 法を用いた連立一次方程式の数値解の精度保証方式について述べる．

3.1 近似逆行列による連立一次方程式の数値解の精度保証

連立一次方程式の数値解の精度保証を行うための原理となるよく知られた定理を以下に示す．

Theorem 1 連立一次方程式 $Ax = b$ の数値解を $\tilde{x}$，A の近似逆行列を R，単位行列を I とする．このとき

$$\|RA - I\|_\infty < 1 \tag{2}$$

ならば，逆行列 A^{-1} が存在し，

$$\|A^{-1}\|_\infty \leq \frac{\|R\|_\infty}{1 - \|RA - I\|_\infty} \tag{3}$$

及び，x^* を真の解として

$$\|x^* - \tilde{x}\|_\infty \leq \frac{\|R\,(b - A\tilde{x})\,\|_\infty}{1 - \|RA - I\|_\infty} \leq \frac{\|R\|_\infty \|b - A\tilde{x}\|_\infty}{1 - \|RA - I\|_\infty} \tag{4}$$

が成り立つ．

すなわち，連立一次方程式 $Ax = b$ に対して，A の近似逆行列 R を求め，真の解の存在の十分条件を上記の定理に基づき検証し，もし存在する場合には，真の解 x^* と数値解 $\tilde{x}$ との間の誤差ノルム $\|x^* - \tilde{x}\|_\infty$ あるいは誤差ベクトル $|x^* - \tilde{x}|$ を評価する．このとき，精度保証の要点は $\|RA - I\|_\infty$ の上限を求めることであり，特に $R \cdot A$ の行列乗算が主計算となる．

誤差ベクトル $|x^* - \tilde{x}|$ についての成分毎評価については，例えば，山本の定理 [4] などを使うとさらにシャープな誤差限界が求まることが分かっている．

3.2 拡張 Strassen 法を用いた精度保証の手順

拡張 Strassen 法を用いた連立一次方程式 $Ax = b$ の数値解の精度保証は以下のような手順となる．

i) 拡張 Strassen 法を適用したブロック LU 分解によって $Ax = b$ を解く．

ii) i) の LU 分解の結果を用いて近似逆行列 R を求める．

iii) 拡張 Strassen 法を $\|RA - I\|_\infty$ の上限の計算に適用し，A の正則性を検証する．

iv) iii) が成功した場合は，誤差ノルム $\|x^*-\tilde{x}\|_\infty$ あるいは誤差ベクトル $|x^*-\tilde{x}|$ の上限を計算する．

ここで，i) については本講究録の古井・鈴木・後の文献 [8] を参照されたい．以下では，ii) 以降について言及する．

4 正則性の検証

いま，i) によって $PA \approx LU$ を満たすような下三角行列 L，上三角行列 U，置換行列 P が計算されているとすると，A の近似逆行列 R は行列方程式

$$AX = I \tag{5}$$

の解として L, U, P を用いて求めることができる．このようにして求めた R に対して，$\|RA-I\|_\infty$ の上限の計算をする．すなわち，

$$\underline{G} \le RA - I \le \overline{G} \iff \text{すべての } (i,j) \text{ に対し} \quad \underline{G_{ij}} \le (RA-I)_{ij} \le \overline{G_{ij}}$$

となるような行列 $\underline{G}, \overline{G}$ を求める必要がある．そこで，荻田の提案した Strassen 法による行列乗算の包み込みアルゴリズム [7] を拡張して，$R \cdot A$ の包み込み計算に拡張 Strassen 法を適用することを考える．

4.1 高速アルゴリズム

Strassen 法や拡張 Strassen 法に，丸めモード制御演算方式による従来の行列乗算の包み込み方法をそのまま適用しようとすると問題が発生する．例えば，Algorithm 1: 3 行目の U_i の計算

$$U_i = A_{i1} * (B_{1s} + B_{2s})$$

を考えると，丸めモード制御演算によって $B_{1s} + B_{2s}$ の真値を区間行列で包み込むことができるが，そのため $A_{i1} * (B_{1s} + B_{2s})$ は，点行列と区間行列の乗算となる．従来の方式では，点行列と区間行列の乗算 (区間行列と点行列の乗算も同様) の包み込み及び区間行列同士の乗算の包み込みに必要な計算量は，それぞれ点行列同士の乗算の近似計算に必要な計算量の約 3 倍及び約 4 倍である．したがって，精度保証に Strassen 法や拡張 Strassen 法を用いると逆に計算量が多くなってしまう．

そこで，本報告では，Strassen の方法に基づいた行列乗算の包み込みを高速に計算する方法 [7] を拡張して，拡張 Strassen 法に基づく方法を提案する．

そのため，まず，非負行列同士の乗算の上限を高速に計算する方式について簡潔に述べる．$X = (x_{ij})$ を $m \times p$ 非負行列，$Y = (y_{ij})$ を $p \times n$ 非負行列とすると，以下の不等式が成り立つ：

$$(XY)_{ij} = \sum_{k=1}^{p} x_{ik} \cdot y_{kj} \le \sum_{k=1}^{p} x_{ik} \left(\max_{1 \le q \le n} y_{kq} \right) = s_{ij} \tag{6}$$

あるいは

$$(XY)_{ij} = \sum_{k=1}^{p} x_{ik} \cdot y_{kj} \le \sum_{k=1}^{p} \left(\max_{1 \le q \le m} x_{qk} \right) y_{kj} = t_{ij}. \tag{7}$$

式 (6), (7) から，

$$(XY)_{ij} \le \min\{s_{ij}, t_{ij}\} \tag{8}$$

が成り立つ．

以上に事実に基づき，区間行列を含む行列乗算の包み込みを高速に求める以下のアルゴリズムが得られる：

$[\underline{D}, \overline{D}] = \mathbf{FastInRePI}(B, \underline{C}, \overline{C})$ ：点行列と区間行列の高速乗算
$[\underline{D}, \overline{D}] = \mathbf{FastInReIP}(\underline{B}, \overline{B}, C)$ ：区間行列と点行列の高速乗算
$[\underline{D}, \overline{D}] = \mathbf{FastInReII}(\underline{B}, \overline{B}, \underline{C}, \overline{C})$ ：区間行列と区間行列の高速乗算

アルゴリズムの詳細は文献 [7] を参照されたい．これらのアルゴリズムに必要な計算量は，点行列同士の乗算の近似計算を 2 回実行するのに必要な計算量とほぼ同じである．

以上の議論に基づき，拡張 Strassen 法を用いて行列乗算を包み込む高速なアルゴリズムを以下に示す．ここで，n は行列サイズを，N は分割数を表すものとする．

Algorithm 2 拡張 *Strassen* 法による行列乗算 $B \cdot C$ の高速な包み込み：

function $[\underline{D}, \overline{D}] = \mathbf{ExtStrassIn}\,(B, C, N)$
$m = n/N;\quad \alpha = 1 : n/2;\quad \beta = n/2 + 1 : n;$
for $p = 1 : N - 3$
  $q = N - p + 1;\quad i = (p-1) * m + 1 : p * m;\quad s = (q-1) * m + 1 : q * m;$
  setround$(+1)$; $\overline{B_1} = B_{i\beta} - B_{i\alpha};\quad \overline{C_1} = C_{\alpha s} + C_{\beta s};$
  setround(-1); $\underline{B_1} = B_{i\beta} - B_{i\alpha};\quad \underline{C_1} = C_{\alpha s} + C_{\beta s};$
  $[\underline{U_p}, \overline{U_p}] = \mathbf{FastInRePI}\,(B_{i\alpha}, \underline{C_1}, \overline{C_1})$;
  $[\underline{V_p}, \overline{V_p}] = \mathbf{FastInReIP}\,(\underline{B_1}, \overline{B_1}, C_{\beta s})$
  setround$(+1)$; $\overline{D_{is}} = \overline{U_p} + \overline{V_p};$
  setround(-1); $\underline{D_{is}} = \underline{U_p} + \underline{V_p};$
end
for $q = N - 1 : -1 : 1$
  if $q == 1$
    $p = N;$
  else
    $p = N - q + 2 * \mathbf{mod}\,(q, 2)$;
  end
  $k = (p-1) * m + 1 : p * m;\quad j = (q-1) * m + 1 : q * m;$
  setround$(+1)$; $\overline{B_2} = B_{k\alpha} - B_{k\beta};\quad \overline{C_2} = C_{\alpha j} + C_{\beta j};$
  setround(-1); $\underline{B_2} = B_{k\alpha} - B_{k\beta};\quad \underline{C_2} = C_{\alpha j} + C_{\beta j};$

$[\underline{L}, \overline{L}] =$ **FastInReIP** $(\underline{B_2}, \overline{B_2}, C_{\alpha j})$;
$[\underline{M}, \overline{M}] =$ **FastInRePI** $(B_{k\beta}, \underline{C_2}, \overline{C_2})$;
setround$(+1)$; $\overline{D_{kj}} = \overline{L} + \overline{M}$;
setround(-1); $\underline{D_{kj}} = \underline{L} + \underline{M}$;
for $u = 1 : N - q -$ **mod** $(q-1, 2)$
  $r = N - u + 1$; $i = (u-1) * m + 1 : u * m$; $s = (r-1) * m + 1 : r * m$;
  if $u \geq N - 2$
    $t = 1$;
    setround$(+1)$; $\overline{B_3} = B_{i\beta} - B_{i\alpha}$; $\overline{C_3} = C_{\alpha s} + C_{\beta s}$;
    setround(-1); $\underline{B_3} = B_{i\beta} - B_{i\alpha}$; $\underline{C_3} = C_{\alpha s} + C_{\beta s}$;
    $[\underline{U_1}, \overline{U_1}] =$ **FastInRePI** $(B_{i\alpha}, \underline{C_3}, \overline{C_3})$;
    $[\underline{V_1}, \overline{V_1}] =$ **FastInReIP** $(\underline{B_3}, \overline{B_3}, C_{\beta s})$;
    setround$(+1)$; $\overline{D_{is}} = \overline{U_1} + \overline{V_1}$;
    setround(-1); $\underline{D_{is}} = \underline{U_1} + \underline{V_1}$;
  else
    $t = u$;
  end
  setround$(+1)$;
  $\overline{B_4} = B_{i\alpha} + B_{k\beta}$; $\overline{C_4} = C_{\alpha j} + C_{\beta s}$; $\overline{B_5} = B_{i\beta} + B_{k\beta}$; $\overline{C_5} = C_{\beta j} - C_{\beta s}$;
  setround(-1);
  $\underline{B_4} = B_{i\alpha} + B_{k\beta}$; $\underline{C_4} = C_{\alpha j} + C_{\beta s}$; $\underline{B_5} = B_{i\beta} + B_{k\beta}$; $\underline{C_5} = C_{\beta j} - C_{\beta s}$;
  $[\underline{P}, \overline{P}] =$ **FastInReII** $(\underline{B_4}, \overline{B_4}, \underline{C_4}, \overline{C_4})$;
  $[\underline{Q}, \overline{Q}] =$ **FastInReII** $(\underline{B_5}, \overline{B_5}, \underline{C_5}, \overline{C_5})$;
  setround$(+1)$; $\overline{D_{ij}} = \overline{P} + \overline{Q} - \underline{M} + \overline{V_t}$;
  $\overline{B_6} = B_{i\alpha} + B_{k\alpha}$; $\overline{C_6} = C_{\alpha s} - C_{\alpha j}$;
  setround(-1); $\underline{D_{ij}} = \underline{P} + \underline{Q} - \overline{M} + \underline{V_t}$;
  $\underline{B_6} = B_{i\alpha} + B_{k\alpha}$; $\underline{C_6} = C_{\alpha s} - C_{\alpha j}$;
  $[\underline{Q}, \overline{Q}] =$ **FastInReII** $(\underline{B_6}, \overline{B_6}, \underline{C_6}, \overline{C_6})$;
  setround$(+1)$; $\overline{D_{ks}} = \overline{P} + \overline{Q} + \overline{L} - \underline{U_t}$;
  setround(-1); $\underline{D_{ks}} = \underline{P} + \underline{Q} + \underline{L} - \overline{U_t}$;
end
end

ここで，命令 **setround(+1)** は丸めモードを「下への丸め」に変更し，**setround(−1)** は丸めモードを「上への丸め」に変更する．また，**setround(0)** でデフォルトの「最近点への丸め」に戻す．丸めモードが一度変更されると，次に丸めモードを変更する **setround** 命令が現れるまで，変更された丸めモードが持続するものと仮定する．丸めモード制御演算方式の詳細については，文献 [6] を参照されたい．

Algorithm 2 により，拡張 Strassen 法を用いて $R \cdot A$ の高速な包み込みができるようになったので，係数行列 A の正則性を検証する以下の Algorithm 3 により，$\|RA - I\|_\infty$ の精度保証を行う．Algorithm 3 の中で，**'usual'** は「通常計算」，**'exstras'** は「拡張 Strassen 法」を

表すものとする.

Algorithm 3 $\|RA - I\|_\infty$ の精度保証：

$$
\begin{array}{l}
\textbf{function } d = \textbf{NormInv}\,(A, R, mode, N) \\
\textbf{switch } mode \\
\textbf{case } \textbf{'usual'} \\
\quad \textbf{setround}(-1);\ \ \underline{G} = R * A; \\
\quad \textbf{setround}(+1);\ \ \overline{G} = R * A; \\
\textbf{case } \textbf{'exstras'} \\
\quad [\underline{G}, \overline{G}] = \textbf{ExtStrassIn}(R, A, N); \\
\textbf{end} \\
\textbf{setround}(-1);\quad \underline{G} = \underline{G} - I; \\
\textbf{setround}(+1);\quad \overline{G} = \overline{G} - I; \\
\overline{G} = \textbf{max}(\textbf{abs}(\underline{G}), \textbf{abs}(\overline{G})); \\
d = \textbf{norm}(\overline{G}, \mathrm{inf});
\end{array}
$$

4.2 数値解の誤差評価

係数行列 A の正則性を保証できると，式 (3), (4) から誤差ノルム $\|x^* - \tilde{x}\|_\infty$ の評価が可能である．そこで，誤差ノルム $\|x^* - \tilde{x}\|_\infty$ の評価を行なうアルゴリズムを以下に示す.

Algorithm 4 誤差ノルム $\|x^* - \tilde{x}\|_\infty$ の精度保証：

$$
\begin{array}{l}
\textbf{function } err = \textbf{NormError}\,(A, b, R, \tilde{x}, mode, N) \\
d = \textbf{NormInv}\,(A, R, mode, N)\,; \\
\textbf{if } \ d < 1 \\
\quad \textbf{setround}(-1);\quad \underline{r} = A * x - b; \\
\quad \textbf{setround}(+1);\quad \overline{r} = A * x - b; \\
\quad \overline{r} = \textbf{max}\,(\textbf{abs}\,(\underline{r})\,, \textbf{abs}\,(\overline{r}))\,; \\
\quad r_{norm} = \textbf{norm}\,(\overline{r}, \mathrm{inf})\,; \\
\quad R_{norm} = \textbf{norm}\,(R, \mathrm{inf})\,; \\
\quad R_{norm} = R_{norm} / -(d-1); \\
\quad err = R_{norm} * r_{norm}; \\
\textbf{else} \\
\quad disp(\text{'}\textit{verification failed.}\text{'}) \\
\textbf{end}
\end{array}
$$

5 数値実験

3.1 節でも述べたように，$\|RA - I\|_\infty$ の精度保証が成功して A の正則性が保証できれば，拡張 Strassen 法を用いた連立一次方程式の精度保証が行えることになる．そこで，まず

$\|RA-I\|_\infty$ の精度保証結果を示した後，誤差ノルム $\|x^*-\tilde{x}\|_\infty$ の精度保証結果を示すことにする．

数値実験は，拡張 Strassen 法における行列の分割数 N を変化させて $\|RA-I\|_\infty$ 及び誤差ノルム $\|x^*-\tilde{x}\|_\infty$ の精度保証を行ったものである．拡張 Strassen 法は通常の Strassen 法と同様，再帰的に適用可能だが，ここでは 1 回のみ適用することにする．数値実験結果として，行列サイズ n が 2 のべき乗，特に，$n = 256, 512, 1024$ の場合の結果を示し，また，A は一様乱数行列とする．実験結果は，数値実験を 10 回行った平均の値とする．

5.1 $\|RA-I\|_\infty$ の評価結果

以下に，$\|RA-I\|_\infty \le d$ の評価結果を示す．表中の usual は拡張 Strassen 法を用いない通常計算による結果を表すものとする．尚，$N=2$ のときは，拡張 Strassen 法は通常の Strassen 法と一致する．

表 1: $n = 256$

	d
usual	1.38×10^{-11}
$N=2$	3.68×10^{-11}
$N=4$	2.83×10^{-11}
$N=8$	2.94×10^{-11}
$N=16$	3.22×10^{-11}
$N=32$	3.33×10^{-11}
$N=64$	3.26×10^{-11}

表 2: $n = 512$

	d
usual	8.09×10^{-11}
$N=2$	2.06×10^{-10}
$N=4$	1.47×10^{-10}
$N=8$	1.55×10^{-10}
$N=16$	1.70×10^{-10}
$N=32$	1.73×10^{-10}
$N=64$	1.81×10^{-10}

表 3: $n = 1024$

	d
usual	1.56×10^{-10}
$N=2$	3.49×10^{-10}
$N=4$	2.98×10^{-10}
$N=8$	3.07×10^{-10}
$N=16$	3.10×10^{-10}
$N=32$	3.16×10^{-10}
$N=64$	3.28×10^{-10}

計算結果から，行列サイズ n の大きさによって多少の差はあるが，通常方式と拡張 Strassen 法を用いた場合には，精度が 1.5 倍から 3 倍程度の差しかなく，ほぼ同等の精度が保証されていると言える．

5.2 誤差ノルム $\|x^*-\tilde{x}\|_\infty$ の評価

次に，誤差ノルム $\|x^*-\tilde{x}\|_\infty \le err$ の評価を以下に示す．

これらの表から，分割数 N を変化させても誤差ノルム $\|x^*-\tilde{x}\|_\infty$ の評価にはまったくと言って良いほど差がなく，同等の結果が得られていることがわかる．これは，前節で示したように，$\|RA-I\|_\infty$ の評価が 1.5 倍から 3 倍程度の差であったことから，式 (4)

$$\|A^{-1}\|_\infty \le \frac{\|R\|_\infty}{1-\|RA-I\|_\infty}$$

の分母の形から考えると，$\|RA-I\|_\infty$ の評価が 1 に近づかない限り，誤差ノルム $\|x^*-\tilde{x}\|_\infty$ の評価にはほとんど影響が表れないことを示している．

表 4: $n = 256$

	err
usual	1.43×10^{-11}
$N = 2$	1.43×10^{-11}
$N = 4$	1.43×10^{-11}
$N = 8$	1.43×10^{-11}
$N = 16$	1.43×10^{-11}
$N = 32$	1.43×10^{-11}
$N = 64$	1.43×10^{-11}

表 5: $n = 512$

	err
usual	2.56×10^{-10}
$N = 2$	2.56×10^{-10}
$N = 4$	2.56×10^{-10}
$N = 8$	2.56×10^{-10}
$N = 16$	2.56×10^{-10}
$N = 32$	2.56×10^{-10}
$N = 64$	2.56×10^{-10}

表 6: $n = 1024$

	err
usual	4.35×10^{-9}
$N = 2$	4.35×10^{-9}
$N = 4$	4.35×10^{-9}
$N = 8$	4.35×10^{-9}
$N = 16$	4.35×10^{-9}
$N = 32$	4.35×10^{-9}
$N = 64$	4.35×10^{-9}

6 おわりに

本報告では，拡張 Strassen 法を用いた連立一次方程式の数値解の精度保証法を提案した．これによって，加算と乗算だけでなく丸めの処理を施す際に問題となる減算を含んだ拡張 Strassen 法を利用しているにも関わらず，提案方式によって通常方式と同等の結果で数値解の精度保証をすることができた．これは，演算量が理論上 3/4 まで削減できる拡張 Strassen 法の応用範囲を広げることができたという意味で有意義な結果であると思われる．

尚，本報告では計算時間について言及していないため，計算時間については今後の課題として取り組んでいきたい．

参考文献

[1] G.H. Golub, C.F. van Loan: *Matrix Computations, 3rd ed.*, The Johns Hopkins University Press: Baltimore and London, 1996.

[2] S. Oishi, S.M. Rump: Fast verification of solutions of matrix equations, *Numer. Math.*, **90** (2002), pp. 755–773.

[3] V. Strassen: Gaussian elimination is not optimal, *Numer. Math.*, **13** (1969), pp. 354–356.

[4] T. Yamamoto: Error bounds for approximate solutions of systems of linear equations, *Japan J. Appl. Math.*, **1**:1 (1984), pp. 157–171.

[5] 後保範: 行列乗算におけるストラッセンの方法の拡張, 京大数理研講究録, **1040** (1998), pp. 61–69.

[6] 大石進一: 精度保証付き数値計算, コロナ社, 2000.

[7] 荻田武史, 大石進一, 後保範: Strassen のアルゴリズムによる行列乗算の高速精度保証, 京大数理研講究録, **1320** (2003), pp. 151–161.

[8] 古井充, 鈴木健二, 後保範: 拡張ストラッセン法の連立一次方程式への適用, 本講究録.