# 大規模連立一次方程式のための高速精度保証法

荻田武史[†,††] 大石進一[††]

大規模密行列を係数行列とする連立一次方程式を計算機上で解いた場合に得られる近似解と厳密解との定量的誤差評価について考える．本論文では係数行列が数万次元程度の問題を取り扱う．これにより，小規模な問題を扱っているときには見えなかった従来の精度保証方式の限界を明らかにするとともに，大規模な問題に適応した新しい精度保証方式を提案する．数値実験によって，大規模な問題に対して提案方式を用いると近似解の計算時間の約 3 倍以内でその精度保証が可能であることを示す．

# Fast Verification Method for Large-scale Linear Systems

Takeshi Ogita[†,††] and Shin'ichi Oishi[††]

This paper is concerned with the problem of verifying the accuracy of the approximate solutions of large-scale dense linear systems. In this paper, the guaranteed error bounds on computed solutions of large-scale linear systems are calculated. Results of numerical experiments have elucidated the limit of applicability of the previous verification methods for large-scale problems under certain conditions. To overcome this, a new verification method is proposed for large-scale problems. Finally, the performance of the proposed method is evaluated.

## 1. はじめに

本論文では，連立一次方程式

$$A\boldsymbol{x} = \boldsymbol{b} \tag{1}$$

に対し，その解の検証について考える．ただし，$A$ は $n \times n$ 実行列，$\boldsymbol{b}$ は $n$ 次の実ベクトルである．理工学における諸問題を数値計算によって解く場合，対象とする問題を数理モデル化し，それを離散化し，最終的に連立一次方程式の求解問題に帰着することが多い．たとえば，偏微分方程式の初期値・境界値問題としてモデル化された問題を境界要素法[14] によって離散化すると，係数行列 $A$ が密行列であるような連立一次方程式が得られ，それは境界面を細かく離散化するほど大規模な問題となる．

このようにして得られた連立一次方程式は，直接解法や反復解法によって計算機上で解くことができるが，一般的に，数値計算によって得られた解と実現象または理論値との間には誤差が含まれるため，その誤差を見積もることは数理モデル化を考えるうえでも重要である．誤差の要因には大きく分けて以下の 3 つがある．

- モデル化誤差
- 離散化誤差
- 計算機による丸め誤差

また，反復解法を用いる場合はこのほかに打切り誤差も発生する．本論文では，式 (1) に直接解法を用いた場合の計算機による丸め誤差について議論する．

通常，式 (1) に対して，計算機を使って得られた解は丸め誤差を含むため厳密解 $\boldsymbol{x}^* := A^{-1}\boldsymbol{b}$ ではなく近似解である（これを数値解と呼ぶ）．近年，精度保証付き数値計算の成果として，$A$ の正則性を検証し，式 (1) の厳密解 $\boldsymbol{x}^*$ と数値解 $\widetilde{\boldsymbol{x}}$ の間に生じる誤差限界

$$\|\widetilde{\boldsymbol{x}} - \boldsymbol{x}^*\|_\infty \le \epsilon_{\mathrm{norm}} \quad \text{（ノルム評価）} \tag{2}$$

あるいは

$$|\widetilde{\boldsymbol{x}} - \boldsymbol{x}^*| \le \boldsymbol{d}_{\mathrm{abs}} \quad \text{（成分ごと評価）} \tag{3}$$

をできるだけ過大評価することなく計算するための様々な方法が開発されてきた[7)~9),12)]．特に，Oishi と Rump によって提案された高速精度保証法[7] では，数値解を得るのと同じ演算量で数値解の精度保証ができる．Oishi-Rump の方法では，IEEE 754 規格に基づく浮動小数点数の丸めモード制御演算方式[6] を利用しているため，非常に高速な精度保証が可能となる．一方，分散並列処理による大規模な精度保証付き数値計算は，現在ではまだほとんど行われていない．

そこで，本論文では係数行列が数万次元程度の大規

† 科学技術振興機構（JST）
Japan Science and Technology Agency
†† 早稲田大学理工学術院
Faculty of Science and Engineering, Waseda University

模密行列であるような連立一次方程式の数値解の精度保証を行う．これにより，比較的小規模な問題を考えていたときには見えなかった従来の精度保証方式の限界を明らかにし，さらに，大規模問題に適応した新しい精度保証方式を提案する．ただし，浮動小数点演算においてオーバフローやアンダフローは起きないものと仮定する．また，数値実験では PC クラスタによる分散並列計算環境において ScaLAPACK [1] を用いる．

本論文の構成は以下のとおりである．2 章では，浮動小数点数の丸めモード制御演算方式について概説する．3 章では，従来の精度保証方式について述べ，さらに，8 ノードの PC クラスタ上で 1 万次元までの密行列に対して数値実験を行い，従来方式の問題点を明らかにする．4 章では，大規模な密行列向きの新しい精度保証方式を提案し，8 ノードおよび 100 ノードの PC クラスタを用いた数値実験によってその性能を評価する．

## 2. 丸めモード制御演算方式

まず，本論文で扱う浮動小数点演算と丸めモード制御演算方式について簡潔に説明する．$\mathbb{F}$ を浮動小数点数の集合とする．本論文で扱う浮動小数点数システムは次の仮定を満たすものとする．まず，$\mathbb{F}$ は対称，すなわち，$x \in \mathbb{F} \Rightarrow -x \in \mathbb{F}$ を満たし，$|x|$ は $x \in \mathbb{F}$ について誤差なしで計算できるものとする．

次に，以下の 3 つの丸めモードを定義する．

**(1) 上への丸め** $c \in \mathbb{R}$ を $f \geq c$ を満たす最も小さな浮動小数点数 $f \in \mathbb{F}$ へ丸める．これを $\triangle : \mathbb{R} \to \mathbb{F}$ と表す．

**(2) 下への丸め** $c \in \mathbb{R}$ を $f \leq c$ を満たす最も大きな浮動小数点数 $f \in \mathbb{F}$ へ丸める．これを $\bigtriangledown : \mathbb{R} \to \mathbb{F}$ で表す．

**(3) 最近点への丸め** $c \in \mathbb{R}$ を $|f - c|$ が最小となる浮動小数点数 $f \in \mathbb{F}$ へ丸める．これを $\Box : \mathbb{R} \to \mathbb{F}$ と表す．

浮動小数点演算が，四則演算子 $\circ \in \{+, -, \cdot, /\}$ と丸め演算子 $\bigcirc \in \{\triangle, \bigtriangledown, \Box\}$ に対して

$$x \circledcirc y = \bigcirc(x \circ y), \quad (\forall x, y \in \mathbb{F}) \tag{4}$$

により定義されるとする．ここで，式 (4) の左辺の $\circledcirc$ は浮動小数点演算を表し，右辺は，通常の実数演算 $\circ$ による結果を浮動小数点数に丸めたものを表す．これらは IEEE 754 規格に従う倍精度浮動小数点システムではすべて満たされている．

ここで，丸めモード制御演算の基本となる行列 $A$ と $B$ の積を包み込む方法[6] を示す．ただし，$A, B \in \mathbb{F}^{n \times n}$ とする．行列積 $A \cdot B$ を「包み込む」というのは，$\underline{C} \leq A \cdot B \leq \overline{C}$ を満たすような下限 $\underline{C}$ と上限 $\overline{C}$ を求めることである．ただし，不等式は成分ごとに成立する．本論文では，アルゴリズムを MATLAB のようなプログラムで表現する．

**Algorithm 1** $C = A \cdot B$ の包み込み：

```
setround(−1);   % 下への丸めモードに変更
C = A * B;      % A·B の下限を計算
setround(+1);   % 上への丸めモードに変更
C = A * B;      % A·B の上限を計算
```

□

ここで，**setround**$(-1)$ は下への丸めモードへ切り替える命令を表し，**setround**$(+1)$ は上への丸めモードへ切り替える命令を表す．いったん丸めのモードが切り替えられると次の丸めモードの切替えの命令が現れるまでは同じ丸めモードが保たれると仮定する．つまり，Algorithm 1 において，1 行目で下への丸めモードに切り替えると次に丸めモードを切り替える命令が現れるまで計算はすべて下への丸めモードで実行されるため，式 (4) より 2 行目では $A \cdot B$ の真の値に対して下限を計算することになる．同様に，3 行目で上への丸めモードに切り替えることにより，4 行目では $A \cdot B$ の真の値に対して上限を計算することになる．つまり，$C = A \cdot B$ の近似計算 2 回と丸めモードの切替え 2 回によって $C$ の包み込みが計算できることが分かる．よって，$A, B \in \mathbb{F}^{n \times n}$ のとき，Algorithm 1 の演算量は約 $4n^3$ flops☆である．

このような丸めモード制御演算方式の大きな利点の 1 つは，行列乗算 $A \cdot B$ の近似計算に BLAS [2] や ATLAS [11] などの高速で信頼性の高い数値計算ライブラリをそのまま利用できるところにあり，古典的な区間演算との最大の差異でもある．そして，この発想は並列計算の場合でもまったく同様に有効である．

本論文では，分散並列計算における密行列どうしの行列乗算の実装として，ScaLAPACK [1] のサブルーチン `PDGEMM` を用いる．ScaLAPACK の構成は，**図 1** のようになっている．このとき，MPI（Message-Passing Interface）を用いて実装されたプログラムの中では，丸めモードの変更が各プロセッサに対して適用されるので，逐次計算のときと同様に行列乗算の結果の包み込みが可能となる．

**Remark 1** 本論文では，行列乗算 $A \cdot B$ を計算するとき，Strassen のアルゴリズム[10] などは使っていないと仮定する．たとえば，Strassen のアルゴリズム

☆ floating point operations：本論文では，浮動小数点数の加算・減算・乗算・除算をそれぞれ 1 flop と数える．

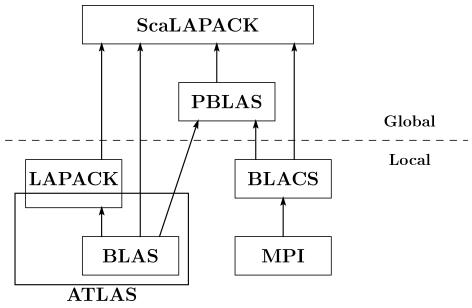


**図 1** ScaLAPACK の構成
Fig. 1 Construction of ScaLAPACK.

を使う場合，Algorithm 1 のように計算しても，必ずしも $A \cdot B$ の包み込みになるとは限らない．詳しくは，文献 13) を参照されたい．また，丸めモード制御の効果がない数値計算ライブラリ（Intel Math Kernel Library など）も存在する． □

**Remark 2** ScaLAPACK は LAPACK の分散並列化版ライブラリであり，内部では LAPACK，BLAS に加えて PBLAS，BLACS，MPI（または PVM）といったライブラリを用いている（図 1）．Remark 1 に関連して，リファレンス版 BLAS や ATLAS を用いる場合はソースコードが公開されているため☆，Algorithm 1 による行列乗算の包み込みが有効であることを確認できるが，ベンダによる製品版などでソースコードが非公開の場合は Algorithm 1 のような方法は不可能である． □

本論文で使う行列・ベクトルに関する定義について記しておく．実行列 $A$ に対して，$|A|$ は $A$ の各要素ごとに絶対値をとった行列を表す．実ベクトル $\boldsymbol{x} = (x_1, \ldots, x_n)^T$ の最大値ノルムは

$$\|\boldsymbol{x}\|_\infty := \max_{1 \le i \le n} |x_i| \tag{5}$$

また，$m \times n$ 実行列 $A = [a_{ij}]$ の最大値ノルムは

$$\|A\|_\infty := \max_{1 \le i \le m} \sum_{j=1}^{n} |a_{ij}| \tag{6}$$

とそれぞれ定義される．また，行列の条件数を

$$\mathrm{cond}(A) := \|A\|_\infty \cdot \|A^{-1}\|_\infty \tag{7}$$

と定義する．

## 3. 従来の精度保証法

本章では，連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ の数値解 $\widetilde{\boldsymbol{x}}$ に対する従来の精度保証法について議論する．まず，基本となる定理[7] を以下に示す．

**Theorem 1** $A \in \mathbb{R}^{n \times n}$，$\boldsymbol{b} \in \mathbb{R}^n$ とする．$\widetilde{\boldsymbol{x}}$ を $A\boldsymbol{x} = \boldsymbol{b}$ の近似解，$R$ を $A$ の近似逆行列，$I$ を $n \times n$ 単位行列とする．このとき

$$\|RA - I\|_\infty < 1 \tag{8}$$

ならば，$A$ は正則で

$$\|A^{-1}\|_\infty \le \frac{\|R\|_\infty}{1 - \|RA - I\|_\infty} \tag{9}$$

$$\|\widetilde{\boldsymbol{x}} - A^{-1}\boldsymbol{b}\|_\infty \le \|A^{-1}\|_\infty \cdot \|A\widetilde{\boldsymbol{x}} - \boldsymbol{b}\|_\infty \tag{10}$$

が成り立つ． □

Theorem 1 によって，係数行列 $A$ の正則性が保証されたときに，式 (10) の右辺の上限を計算することによって，数値解が少なくとも何桁まで正しい精度を持っているかを保証することが可能となる．その中でも，式 (8) の判定が $A$ の正則性を保証するため最も重要であり，また，この精度保証法の主計算となる．$\|RA - I\|_\infty < 1$ を示すことができれば，厳密解に対する数値解の誤差限界を精度良く求める方法[4),5),12)] はすでに知られているので，以下では $\|RA - I\|_\infty$ の上限の計算法について議論する．

**Remark 3** $\|RA - I\|_\infty$ の評価は，真の値よりも多少過大評価となっても最終的な精度保証結果にはほとんど影響を与えない．たとえば，$\|RA - I\|_\infty$ の評価値が 0.5 と比較的大きい場合でも，式 (9) 右辺の分母は $1 - 0.5 = 0.5$ となり，精度保証全体の評価としては 2 倍しか悪くならない．したがって，計算量の点から考えると，$\|RA - I\|_\infty$ の上限を過大評価なく計算することよりも，ある程度過大評価でも（評価値が 1 より小さい範囲であれば）高速に計算することのほうが重要となる． □

また，$A\boldsymbol{x} = \boldsymbol{b}$ の数値解の計算には，$A$ の部分軸交換付 LU 分解を用いることを前提とし，$LU \approx PA$ すなわち $\|PA - LU\|_\infty \ll \|A\|_\infty$ のような $L$，$U$，$P$ が得られているものとする．ここで，$L$ は単位下三角行列，$U$ は上三角行列，$P$ は軸交換にともなう置換行列である．このとき，LU 分解に必要な演算量は $\frac{2}{3}n^3$ flops であり，LU 分解が得られているときに数値解の計算に必要な演算量は $\mathcal{O}(n^2)$ flops である．

### 3.1 近似逆行列を使う方式

最も素朴な方法は，$A$ の近似逆行列 $R$ を明示的に求めてから $RA - I$ を計算する方法[7] である．以下に，近似逆行列を利用して $\|RA - I\|_\infty$ の上限 $\alpha$ を計算するアルゴリズムを示す．

**Algorithm 2** 近似逆行列を使った $\|RA - I\|_\infty$ の上限 $\alpha$ の計算：

☆ たとえば，http://netlib.org/

**function** $\alpha = \mathbf{vinv}(A, L, U, P)$
$X_U = I/U$; % Solve $X_U U = I$ for $X_U$
$R = X_U/L$; % Solve $RL = X_U$ for $R$
$R = R * P$; % $R \approx A^{-1}$
**setround**$(-1)$;
$\underline{G} = R * A - I$;
**setround**$(+1)$;
$\overline{G} = R * A - I$;
$\overline{G} = \mathbf{max}(|\underline{G}|, |\overline{G}|)$; % $\max\{|\underline{G}_{ij}|, |\overline{G}_{ij}|\}$
$\alpha = \mathbf{norm}(\overline{G}, \mathrm{inf})$; % $\beta \geq \|\overline{G}\|_\infty$

□

最初の 2 行目から 4 行目までは，LAPACK などで用いられているような，LU 分解の結果から逆行列を求める標準的な方法である．ただし，実際には置換行列 $P$ を 2 次元配列として保持する必要はないことに注意する．ScaLAPACK を用いて実装する場合は，近似逆行列 $R$ を求めるためにサブルーチン `PDGETRI` が利用可能であり，その演算量は $\frac{4}{3}n^3$ flops である．また，$\underline{G}$, $\overline{G}$ の計算にはサブルーチン `PDGEMM` を用い，演算量はそれぞれ $2n^3$ flops である．よって，Algorithm 2 全体の演算量は $\frac{16}{3}n^3$ flops となり，これは LU 分解に必要な演算量の 8 倍である．

### 3.2 LU 分解の事前誤差評価を使う方式

次に，LU 分解の事前誤差解析を使って $\|RA - I\|_\infty$ を評価する定理[7] を示す．

**Theorem 2** $A \in \mathbb{R}^{n\times n}$ とする．$A$ の LU 分解の近似を $L$, $U$，その逆行列の近似を $X_L$, $X_U$ とする．そのとき

$$\|RA - I\|_\infty \leq 2\gamma_n \||X_U|(|X_L|(|L|(|U|\boldsymbol{e})))\|_\infty + \gamma_n \||X_U|(|U|\boldsymbol{e})\|_\infty \qquad (11)$$

が成立する．ただし，$\boldsymbol{e} := (1, \ldots, 1)^T$ および

$$\gamma_n := \frac{n\epsilon}{1 - n\epsilon}$$

で，$\epsilon$ は浮動小数点相対精度☆である． □

**Remark 4** 厳密には，LU 分解の近似を求める方法にある程度の制限があるが，これ以上ここでは言及しない（少なくとも，BLAS や LAPACK のリファレンス版ではこれを満たしている）．詳しくは，文献 7) を参照されたい． □

Theorem 2 に基づき，LU 分解の事前誤差評価を利用して $\|RA - I\|_\infty$ の上限 $\alpha$ を計算するアルゴリズムを示す．

**Algorithm 3** LU 分解の事前誤差評価を使った $\|RA - I\|_\infty$ の上限 $\alpha$ の計算：

**function** $[\alpha, X_L, X_U] = \mathbf{vlu}(A, L, U, P)$
$X_U = I/U$; % Solve $X_U U = I$ for $X_U$
$X_L = I/L$; % Solve $X_L L = I$ for $X_L$
$\boldsymbol{e} = \mathbf{ones}(n, 1)$; % $\boldsymbol{e} = (1, \ldots, 1)^T$
**setround**$(+1)$;
$\boldsymbol{s} = |X_U| * (|X_L| * (|L| * (|U| * \boldsymbol{e})))$;
$\boldsymbol{t} = |X_U| * (|U| * \boldsymbol{e})$;
$\gamma = n * \epsilon/(1 - n * \epsilon)$;
$\alpha = \gamma * (2 * \mathbf{norm}(\boldsymbol{s}, \mathrm{inf}) + \mathbf{norm}(\boldsymbol{t}, \mathrm{inf}))$;

□

ここで，8 行目の $\gamma$ の計算では除算の部分のみ丸め誤差が発生することに注意する．ScaLAPACK を使って実装する場合は，$L$ の近似逆行列 $X_L$ と $U$ の近似逆行列 $X_U$ を求めるためにサブルーチン `PDTRTRI` が利用可能であり，その演算量はそれぞれ $\frac{1}{3}n^3$ flops である．よって，Algorithm 3 全体の演算量は，$\frac{2}{3}n^3$ flops となり，LU 分解と同じ演算量で数値解の精度保証ができる．

一般的に，$\mathrm{cond}(L) \approx \mathcal{O}(n)$ および $\mathrm{cond}(U) \approx \mathrm{cond}(A)$ が成り立つので，Theorem 2 による評価は式 (11) から $\mathcal{O}(n^2\epsilon) \cdot \mathrm{cond}(A)$ となることが分かる．したがって，評価値が $n$ の二乗のオーダで増加するため，問題サイズ ($n$) が大きくなると Algorithm 3 の適用が困難となることが予測される．

### 3.3 数値実験 1

ScaLAPACK を用いて，これまでに示した 2 つの精度保証方式を PC クラスタに実装する．数値実験の計算環境は，PC 8 台を 100 Mbps の LAN で接続した PC クラスタである．計算環境の詳細は以下のとおりである．

- Platform：Dell PowerEdge 1550
- CPU：1 GHz Intel Pentium 3 プロセッサ（二次キャッシュ 256 KB）
- メモリ：640 MB（128 MB + 256 MB × 2）
- ネットワークトポロジ：スター型配線（PCi Fast Ethernet Switch FX-16NH）
- OS：Red Hat Linux 7.1
- コンパイラ：gcc，g77 version 2.95.3 20010315
- ライブラリ：ScaLAPACK 1.7，LAPACK 3.0，ATLAS 3.4.2，MPICH 1.2.5，BLACS 1.1

$A \in \mathbb{F}^{n\times n}$, $\boldsymbol{b} \in \mathbb{F}^n$ として，連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ の数値解の計算とその精度保証について考える．係数行列 $A$ には，ScaLAPACK のテストプログラムに付属している関数 `PDMATGEN` でランダム行列を作成し（$A$ の要素は区間 $[-1, 1]$ の一様乱数），右辺ベクトル $\boldsymbol{b}$ の要素はすべて 1 にした．問題サイズ $n$ は，2,000

☆ IEEE 754 倍精度浮動小数点システムの場合は，$\epsilon := 2^{-53}$.

**表 1** $\|RA-I\|_\infty$ の評価：8CPU
Table 1 Evaluation of $\|RA-I\|_\infty$: 8CPU.

| $n$ | 方式 INV | 方式 LU |
|---|---|---|
| 2,000 | $2.94\times10^{-10}$ | $8.33\times10^{-3}$ |
| 4,000 | $2.56\times10^{-10}$ | $1.80\times10^{-1}$ |
| 6,000 | $6.32\times10^{-10}$ | 1.16 ($>1$) |
| 8,000 | $2.89\times10^{-9}$ | — |
| 10,000 | $6.54\times10^{-9}$ | — |

— 計算せず

**表 2** 数値解とその精度保証の計算時間（秒）：8CPU
Table 2 Elapsed time [sec] for calculating numerical solutions and their verifications: 8CPU.

| $n$ | 数値解 | 方式 INV | 方式 LU |
|---|---|---|---|
| 2,000 | 7.93 | 24.8 (3.1) | 6.93 (0.87) |
| 4,000 | 35.6 | 138 (3.9) | 33.2 (0.93) |
| 6,000 | 79.5 | 395 (5.0) | * |
| 8,000 | 167 | 882 (5.3) | — |
| 10,000 | 270 | 1,565 (5.8) | — |

* 精度保証失敗 — 計算せず

から 10,000 とする．このとき，$A$ の条件数は $10^4$ から $10^5$ くらいのオーダであり，それほど悪条件の問題ではないことに注意する．

まず，**表 1** にそれぞれの方式による $\|RA-I\|_\infty$ の評価結果を示す．この値が 1 より小さいことがいえれば，Theorem 1 より精度保証が可能となる．今後，以下のように 2 つの精度保証方式を表記する．

- **方式 INV**：$A$ の近似逆行列を用いる方式（Algorithm 2）
- **方式 LU**：LU 分解やその近似逆行列の事前誤差評価を用いる方式（Algorithm 3）

表 1 から，方式 INV を使うと，ランダム行列のような問題に対しては大きな次元数でも精度保証が可能であることが分かる．一方，方式 LU では，次元数が 6,000 以上になると $\|RA-I\|_\infty$ を評価した値が 1 を超えるため，精度保証に失敗する．

次に，**表 2** にそれぞれの方式による数値解とその精度保証の計算時間を示す．表中の括弧内の数字は，数値解の計算時間（LU 分解に要した計算時間も含む）に対する比率を表す．

表 2 から，精度保証が成功している範囲では，方式 LU が非常に高速であることが分かる．一方，問題サイズが大きいと，方式 INV では，数値解の計算時間に対して約 6 倍の計算時間を必要としている．

以上の結果から，大規模な問題を扱うときは，従来の精度保証方式では多くの計算時間を必要としたり，あるいは，そのままでは適用できなくなったりする可能性があることが分かった．

## 4. 提 案 方 式

本章では，大規模な問題にも適用可能な新しい高速な精度保証方式を提案する．

### 4.1 理論とアルゴリズム

方式 LU では $\|RA-I\|_\infty$ の評価に対してすべて事前誤差評価を用いているため，$\|RA-I\|_\infty$ を最も過大評価している．特に式 (11) の右辺第 1 項の評価が支配的であり，問題サイズ $n$ が大きくなると $A$ の条件数がそれほど大きな値でなくても，その評価値は大きな値となってしまう．提案方式の基本的なアイディアは，方式 LU の評価方法を見直し，適度に後退誤差解析による事前誤差評価を用いて $\|RA-I\|_\infty$ の過大評価を抑える方式を考えることである．

最初に，三角行列の近似逆行列の事前誤差評価に関する定理[3] を示す．

**Theorem 3（Higham）** $T\in\mathbb{R}^{n\times n}$ を三角行列，$I$ を $n\times n$ 単位行列，行列方程式 $XT=I$ を $X$ について解いたときの数値解を $\widetilde{X}$ とする．そのとき

$$|\widetilde{X}T-I|\le\gamma_n|\widetilde{X}||T| \tag{12}$$

が成立する． □

この定理から，部分的に事前誤差評価を用いて $|RA-I|$ を評価しなおし，$\|RA-I\|_\infty$ の上限を計算するための定理を以下に示す．

**Theorem 4** $A\in\mathbb{R}^{n\times n}$ とする．$A$ の LU 分解の近似を $L$，$U$，その逆行列の近似を $X_L$，$X_U$ とする．そのとき

$$\begin{aligned}&\|RA-I\|_\infty\\ &\le\||X_U|(|X_LPA-U|+\gamma_n|U|)\|_\infty &(13)\\ &\le\|X_U\|_\infty\cdot\||X_LPA-U|+\gamma_n|U|\|_\infty &(14)\end{aligned}$$

が成立する． □

**証明** $R:=X_UX_LP$ とおくと，

$$\begin{aligned}|RA-I|&=|X_UX_LPA-I|\\&=|X_U(X_LPA-U)+(X_UU-I)|\\&\le|X_U(X_LPA-U)|+|X_UU-I|\end{aligned}$$

であり，Theorem 3 より

$$\begin{aligned}|RA-I|&\le|X_U||X_LPA-U|+\gamma_n|X_U||U|\\&=|X_U|(|X_LPA-U|+\gamma_n|U|)\end{aligned}$$

である．よって，

$$\begin{aligned}&\|RA-I\|_\infty\\&\le\||X_U|(|X_LPA-U|+\gamma_n|U|)\|_\infty\\&\le\|X_U\|_\infty\cdot\||X_LPA-U|+\gamma_n|U|\|_\infty\end{aligned}$$

となる． （証明終り）

この評価方法では，式 (11) の右辺第 1 項のような評価が解消され，問題サイズ $n$ が大きくなってもそ

の評価値がそれほど大きな値にはならないことが期待できる．実際，Theorem 4 による評価は式 (14) から $\mathcal{O}(n\epsilon)\cdot\mathrm{cond}(A)$ となるので，Theorem 2 による評価 $\mathcal{O}(n^2\epsilon)\cdot\mathrm{cond}(A)$ から改善されていることが分かる．

Theorem 4 に基づき，$\|RA-I\|_\infty$ の上限 $\alpha$ を計算するアルゴリズムを示す．

**Algorithm 4** 提案方式による $\|RA-I\|_\infty$ の上限 $\alpha$ の計算：

**function** $\alpha = \mathbf{vlup}(A, L, U, P)$
$X_U = I/U$; % Solve $X_U U = I$ for $X_U$
$X_L = I/L$; % Solve $X_L L = I$ for $X_L$
$B = P * A$; % $B \leftarrow PA$
**setround**$(-1)$;
$\underline{T} = X_L * B - U$;
**setround**$(+1)$;
$\overline{T} = X_L * B - U$;
$\gamma = n*\epsilon/(1-n*\epsilon)$;
$T = \mathbf{max}(|\underline{T}|, |\overline{T}|) + \gamma * |U|$;
$\alpha = \mathbf{norm}(X_U, \mathrm{inf}) * \mathbf{norm}(T, \mathrm{inf})$;

□

ScaLAPACK を使って実装する場合は，Algorithm 3 と同様に，$L$ の近似逆行列 $X_L$ と $U$ の近似逆行列 $X_U$ の計算にサブルーチン PDTRTRI が利用可能であり，その演算量はそれぞれ $\frac{1}{3}n^3$ flops である．また，$\underline{T}$，$\overline{T}$ の計算には三角行列と密行列の積を計算するサブルーチン PDTRMM を用い，演算量はそれぞれ $n^3$ flops である．よって，Algorithm 4 全体の演算量は，$\frac{8}{3}n^3$ flops となる．

ここで，これまで示してきた精度保証方式の演算量と，それらの LU 分解の演算量に対する比を**表 3** にまとめる．ただし，各精度保証方式の演算量には LU 分解に必要な演算量は含まれていない．提案方式を用いると，LU 分解の演算量の 4 倍でその精度保証ができる．この提案方式では，方式 LU の主計算である $X_L$，$X_U$ の計算結果を再利用可能なので，実際には，Algorithm 5 のように最初に高速な方式 LU（Algorithm 3）を実行し，それが失敗した場合（$\alpha \geq 1$）は提案方式（Algorithm 4）を使うようにする，という 2 段階方式が方式 LU よりもロバストかつ方式 INV よりも高速な精度保証法になると考えられる．

**表 3** 各精度保証方式の演算量
Table 3 Computational cost of verification methods.

| 方式 | 演算量 [flops] | LU 分解との比 |
|---|---|---|
| 方式 INV | $\frac{16}{3}n^3$ | 8 |
| 方式 LU | $\frac{2}{3}n^3$ | 1 |
| 提案方式 | $\frac{8}{3}n^3$ | 4 |

**Algorithm 5** 2 段階方式による $\|RA-I\|_\infty$ の上限 $\alpha$ の計算：

**function** $\alpha = \mathbf{vlup2}(A, L, U, P)$
$[\alpha, X_L, X_U] = \mathbf{vlu}(A, L, U, P)$; % 方式 LU
**if** $\alpha < 1$, **return**, **end**
$B = P * A$;
**setround**$(-1)$;
$\underline{T} = X_L * B - U$;
**setround**$(+1)$;
$\overline{T} = X_L * B - U$;
$\gamma = n*\epsilon/(1-n*\epsilon)$;
$T = \mathbf{max}(|\underline{T}|, |\overline{T}|) + \gamma * |U|$;
$\alpha = \mathbf{norm}(X_U, \mathrm{inf}) * \mathbf{norm}(T, \mathrm{inf})$;

□

Algorithm 5 に必要な演算量は行列 $A$ の性質に依存するが，一般的には $A$ の条件数が比較的小さい場合や行列のサイズがそれほど大きくない場合は方式 LU と同じ $\frac{2}{3}n^3$ flops，それ以外の場合は $\frac{8}{3}n^3$ flops となる．

### 4.2 数値実験 2

ここで，3 章で行った数値実験と同じ数値例に対して 2 段階方式（Algorithm 5）を評価する．数値実験の計算環境は 3 章のものと同じである．

**表 4** に 2 段階方式による $\|RA-I\|_\infty$ の評価結果を示す．比較のため，方式 INV による結果も再度，表に載せる．参考のために，**表 5** に数値解の精度保証結果も示しておく．また，**表 6** に 2 段階方式による数値解の精度保証に要した計算時間を示す．表中の括弧内の数字は，数値解の計算時間（LU 分解に要した計算時間も含む）に対する比率を表す．

**表 4** 2 段階方式による $\|RA-I\|_\infty$ の評価：8CPU
Table 4 Evaluation of $\|RA-I\|_\infty$ by a two-stage method: 8CPU.

| $n$ | 方式 INV | 2 段階方式 |
|---|---|---|
| 2,000 | $2.94\times10^{-10}$ | $8.33\times10^{-3}$ |
| 4,000 | $2.56\times10^{-10}$ | $1.80\times10^{-1}$ |
| 6,000 | $6.32\times10^{-10}$ | $5.28\times10^{-6}*$ |
| 8,000 | $2.89\times10^{-9}$ | $9.91\times10^{-6}*$ |
| 10,000 | $6.54\times10^{-9}$ | $1.85\times10^{-5}*$ |

$*$ 方式 LU による精度保証失敗

**表 5** 数値解の精度保証結果 ($\|\tilde{\boldsymbol{x}} - A^{-1}b\|_\infty$)：8CPU
Table 5 Result of verification for $\|\tilde{\boldsymbol{x}} - A^{-1}b\|_\infty$: 8CPU.

| $n$ | 方式 INV | 2 段階方式 |
|---|---|---|
| 2,000 | $8.05\times10^{-10}$ | $7.10\times10^{-9}$ |
| 4,000 | $6.04\times10^{-10}$ | $6.08\times10^{-8}$ |
| 6,000 | $1.48\times10^{-9}$ | $1.59\times10^{-7}*$ |
| 8,000 | $7.19\times10^{-9}$ | $3.99\times10^{-7}*$ |
| 10,000 | $1.45\times10^{-8}$ | $7.76\times10^{-7}*$ |

$*$ 方式 LU による精度保証失敗

**表 6** 2 段階方式による計算時間（秒）：8CPU
Table 6 Elapsed time [sec] by a two-stage method: 8CPU.

| $n$ | 数値解 | 方式 INV | 2 段階方式 |
|---|---|---|---|
| 2,000 | 7.93 | 24.8 (3.1) | 6.93 (0.87) |
| 4,000 | 35.6 | 138 (3.9) | 33.2 (0.93) |
| 6,000 | 79.5 | 395 (5.0) | 268 (3.4) ∗ |
| 8,000 | 167 | 882 (5.3) | 579 (3.5) ∗ |
| 10,000 | 270 | 1,565 (5.8) | 981 (3.6) ∗ |

∗ 方式 LU による精度保証失敗

これらの結果から，$\|RA-I\|_\infty$ の評価については方式 INV よりもある程度は過大評価となるが，方式 INV よりも格段に高速な精度保証が可能であり，さらに，方式 LU よりもロバストであるため，より大規模な問題に適応した方法であることが分かる．

今回，数値解の精度保証に関しては，Theorem 1 に基づき，方式 INV では

$$\|A^{-1}\|_\infty \le \frac{\|R\|_\infty}{1-\|RA-I\|_\infty} \le \overline{\alpha}_1$$
$$\|\widetilde{\boldsymbol{x}} - A^{-1}\boldsymbol{b}\|_\infty \le \overline{\alpha}_1 \cdot \|A\widetilde{\boldsymbol{x}} - \boldsymbol{b}\|_\infty$$

2 段階方式では

$$\|A^{-1}\|_\infty \le \frac{\|X_U\|_\infty \cdot \|X_L\|_\infty}{1-\|X_U X_L PA-I\|_\infty} \le \overline{\alpha}_2$$
$$\|\widetilde{\boldsymbol{x}} - A^{-1}\boldsymbol{b}\|_\infty \le \overline{\alpha}_2 \cdot \|A\widetilde{\boldsymbol{x}} - \boldsymbol{b}\|_\infty$$

という評価式を用いた．そのため，数値解の精度保証結果では，$\|A^{-1}\|_\infty$ の評価値の差から 2 段階方式による結果が方式 INV に比べて過大評価となっている．これについては Yamamoto の定理[12] などを用いることによって，方式 INV による評価も含めて改善することができる．ただし，本論文の主旨から外れるため評価方法の改善についてはこれ以上言及しない．

### 4.3 数値実験 3

次に，係数行列 $A$ の条件数の増加に対する各精度保証方式のロバスト性を調べる．そこで，$n = 1{,}000$ に固定して，$A$ の条件数を 1 から $10^{16}$ まで変化させる（$A$ の要素は区間 $[-1, 1]$ の間の一様乱数）．結果を**図 2** に示す．図中の ‘INV’，‘LU’，‘New’ はそれぞれ方式 INV，方式 LU，提案方式による結果を表している．また，‘True’ は高精度演算を使って求めた正確な $\|RA-I\|_\infty$ の値を表す．

図 2 から，問題サイズ $n$ が 1,000 程度のとき，条件数が，方式 INV では $10^{14}$ 程度まで，方式 LU では $10^7$ 程度まで，提案方式では $10^{11}$ 程度までは精度保証が成功していることが分かる．各精度保証方式による評価は，それぞれ異なる度合いで過大評価となっているが，条件数の増加に対して極端に悪い評価になるわけではなく，過大評価の割合はほぼ一定である．また，2 段階方式（Algorithm 5）の場合は，条件数が $10^7$ 程度までは方式 LU，それ以上の場合は提案方式と同じ評価になる．

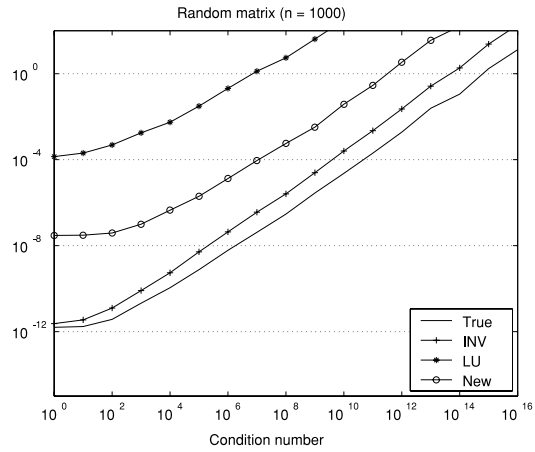


**図 2** 条件数ごとの $\|RA-I\|_\infty$ の評価
Fig. 2 Evaluation of $\|RA-I\|_\infty$ corresponding to condition number.

$A$ の条件数の増加にともなって，各方式の $\|RA-I\|_\infty$ の正確な値自身が増加しているのは，条件数が大きくなると LU 分解自身の精度が落ちるためである．したがって，$\|RA-I\|_\infty$ の厳密な値を小さくするためには $R$ の精度あるいは LU 分解の精度を高くする必要がある．

### 4.4 数値実験 4

最後に，より大規模なシステムに対して各精度保証方式を評価する．そのため，計算環境を変えて，数値実験 1，2 と同様に係数行列がランダム行列であるような問題を解く．数値実験の計算環境は，PC 100 台を 1 Gbps の LAN で接続した PC クラスタである．計算環境の詳細は以下のとおりである．

- Platform：Intel サーバーボード S845WD1
- CPU：2.4 GHz Intel Pentium 4 プロセッサ（二次キャッシュ512 KB）
- メモリ：1 GB（512 MB × 2）
- ネットワークトポロジ：スター型配線（NETGEAR Gigabit Switch GS516T，GS524T）
- OS：Red Hat Linux 7.3
- コンパイラ：gcc，g77 version 2.96 20000731
- ライブラリ：ScaLAPACK 1.7，LAPACK 3.0，ATLAS 3.4.2，MPICH 1.2.5，BLACS 1.1

まず，この計算環境における ScaLAPACK の性能を**図 3** に示す．これは，$n \times n$ 実行列どうしの乗算に対して，`PDGEMM` を用いて $n$ を 5,000 から 45,000 まで

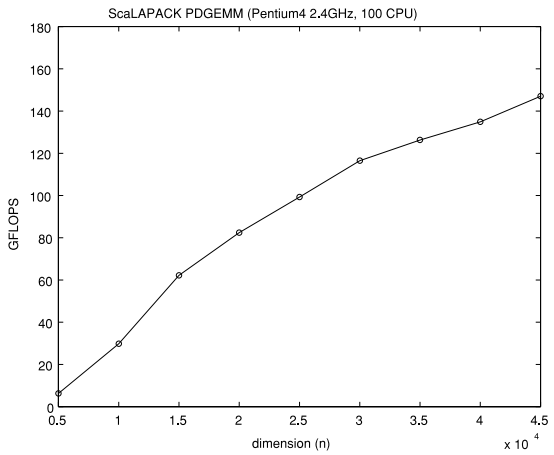


**図 3** ScaLAPACK（PDGEMM）の性能評価（2.4 GHz Pentium 4, 100CPU）
Fig. 3 Performance of PDGEMM in ScaLAPACK (2.4 GHz Pentium 4, 100CPU).

変化させたときの GFLOPS☆を示している．2.4 GHz Pentium 4 の 1CPU では，ATLAS における `DGEMM` の実効ピーク性能が 3 GFLOPS 程度であるので，データ通信のオーバヘッドなどを考慮しなければ，100CPU でのピーク性能は 300 GFLOPS 程度になる．よって，ピーク性能に対して，次元数 $n$ が 1 万のときに 10%程度，2 万のときに 30%程度，3 万のときに 40%程度，4 万のときで 45%程度の性能が出ていることが分かる．

以上のような計算環境において，連立一次方程式の数値解の精度保証を行う．**表 7** に方式 INV および提案方式による $\|RA-I\|_\infty$ の評価結果を示す．参考のため**表 8** に $\|\widetilde{\boldsymbol{x}}-A^{-1}b\|_\infty$ の評価結果をそれぞれ示す．また，**表 9** に数値解の計算時間（LU 分解に要した計算時間も含む），方式 INV および提案方式による精度保証の計算時間を示す．表中の括弧中の数字は，数値解の計算時間に対する比率を表す．

これらの結果から，提案方式は，より大規模な問題にも適用可能であり，方式 INV よりも高速であることが分かる．

## 5. おわりに

大規模な問題を扱う場合は，計算時間や精度の問題だけでなく，メモリの問題も重要になってくる．現時点では，いずれの精度保証方式でも，数値解を計算するだけのときと比べて 3 倍から 5 倍くらいのメモリ量が必要となる．そのため，計算時間をなるべく増加させずに必要なメモリ量を低減するような精度保証方式を開発することが今後の課題である．

☆ Giga floating point operations per second：1 秒間に何十億回浮動小数点演算をしたかを表す．

**表 7** $\|RA-I\|_\infty$ の評価：100CPU
Table 7 Evaluation of $\|RA-I\|_\infty$: 100CPU.

| $n$ | 方式 INV | 提案方式 |
|---|---|---|
| 10,000 | $1.30\times10^{-8}$ | $2.57\times10^{-7}$ |
| 15,000 | $2.07\times10^{-8}$ | $6.96\times10^{-7}$ |
| 20,000 | $1.75\times10^{-8}$ | $1.74\times10^{-6}$ |
| 25,000 | $1.23\times10^{-7}$ | $2.59\times10^{-6}$ |
| 30,000 | $2.34\times10^{-7}$ | $4.03\times10^{-6}$ |

**表 8** 数値解の精度保証結果（$\|\widetilde{\boldsymbol{x}}-A^{-1}b\|_\infty$）：100CPU
Table 8 Result of verification for $\|\widetilde{\boldsymbol{x}}-A^{-1}b\|_\infty$: 100CPU.

| $n$ | 方式 INV | 提案方式 |
|---|---|---|
| 10,000 | $1.06\times10^{-8}$ | $5.63\times10^{-7}$ |
| 15,000 | $1.62\times10^{-8}$ | $2.01\times10^{-6}$ |
| 20,000 | $1.02\times10^{-8}$ | $3.61\times10^{-6}$ |
| 25,000 | $7.27\times10^{-8}$ | $8.22\times10^{-6}$ |
| 30,000 | $1.25\times10^{-7}$ | $1.31\times10^{-5}$ |

**表 9** 数値解とその精度保証の計算時間（秒）：100CPU
Table 9 Elapsed time [sec] for calculating numerical solutions and their verifications: 100CPU.

| $n$ | 数値解 | 方式 INV | 提案方式 |
|---|---|---|---|
| 10,000 | 79.1 | 138 (1.7) | 117 (1.5) |
| 15,000 | 130 | 328 (2.5) | 257 (2.0) |
| 20,000 | 207 | 616 (3.0) | 463 (2.2) |
| 25,000 | 303 | 1,034 (3.4) | 758 (2.5) |
| 30,000 | 408 | 1,595 (3.9) | 1,119 (2.7) |

**謝辞** 本研究は，科学技術振興機構戦略的創造研究推進事業（シミュレーション技術の革新と実用化基盤の構築「数値線形シミュレーションの精度保証に関する研究」），日本学術振興会科学研究費補助金（若手研究（B）No. 16700017）および文部科学省 21 世紀 COE プログラム（Productive ICT Academia Program, 早稲田大学）の補助を受けた．

## 参 考 文 献

1) Blackford, L.S., et al.: *ScaLAPACK Users' Guide*, SIAM, Philadelphia, PA (1997).
2) Dongarra, J.J., et al.: Algorithm 679: A set of Level 3 Basic Linear Algebra Subprograms, *ACM Trans. Math. Softw.*, Vol.16, pp.1–17 (1990).
3) Higham, N.J.: *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, PA (2002).
4) Ogita, T., Oishi, S. and Ushiro, Y.: Computation of Sharp Rigorous Componentwise Error Bounds for the Approximate Solutions of Systems of Linear Equations, *Reliable Computing*, Vol.9, No.3, pp.229–239 (2003).

5) Ogita, T., Oishi, S. and Ushiro, Y.: Fast Inclusion and Residual Iteration for Solutions of Matrix Equations, *Inclusion Methods for Nonlinear Problems* (*Comput. Suppl.* Vol.16), Herzberger, J. (Ed.), pp.171–184, Springer WienNewYork, Austria (2002).
6) Oishi, S.: Fast Enclosure of Matrix Eigenvalues and Singular Values via Rounding Mode Controlled Computation, *Linear Alg. Appl.*, Vol.324, pp.133–146 (2001).
7) Oishi, S. and Rump, S.M.: Fast Verification of Solutions of Matrix Equations, *Numer. Math.*, Vol.90, No.4, pp.755–773 (2002).
8) Rump, S.M.: On the Solution of Interval Linear Systems, *Computing*, Vol.47, pp.337–353 (1992).
9) Rump, S.M.: Verification Methods for Dense and Sparse Systems of Equations, *Topics in Validated Computations*, Herzberger, J. (Ed.), pp.63–135, Elsevier, Amsterdam (1994).
10) Strassen, V.: Gaussian Elimination is Not Optimal, *Numer. Math.*, Vol.13, pp.354–356 (1969).
11) Whaley, R.C., Petitet, A. and Dongarra, J.J.: Automated Empirical Optimization of Software and the ATLAS Project, *Parallel Comput.*, Vol.27, No.1–2, pp.3–25 (2001).
12) Yamamoto, T.: Error Bounds for Approximate Solutions of Systems of Equations, *Japan J. Appl. Math.*, Vol.1, No.1, pp.157–171 (1984).
13) 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，京大数理解析研究所講究録，Vol.1320, pp.151–161 (2003).
14) 神谷紀生，北　栄輔：偏微分方程式の数値解法，工系数学講座，第 11 巻，共立出版 (1998).

(平成 16 年 4 月 5 日受付)
(平成 16 年 7 月 27 日再受付)
(平成 16 年 10 月 22 日採録)

**荻田　武史**

2003 年早稲田大学大学院理工学研究科博士後期課程修了．2003 年同大学大学院理工学研究科客員講師．2003 年同大学小野梓記念学術賞受賞．現在，科学技術振興機構研究員．数値計算の研究に従事．

**大石　進一**

1981 年早稲田大学大学院理工学研究科博士後期課程修了．1989 年同大学教授．電子情報通信学会論文賞（3 回），同学会猪瀬賞，大川出版賞，丹羽記念賞，小野梓賞各受賞．