Vol.2016-HPC-157 No.1
2016/12/21

大規模並列計算機における連立一次方程式の精度保証付き数値計算に対する性能評価

森倉 悠介[1,a)]　椋木 大地[2]　深谷 猛[3]　山中 脩也[4]　大石 進一[1]

概要：本研究では，大規模分散並列計算機における，係数行列を密とする連立一次方程式の精度保証付き数値計算の性能評価を行う．連立一次方程式の精度保証付き数値計算とは，近似解を求めることに加え，真の解と近似解との誤差上限を厳密に求める計算法である．連立一次方程式の精度保証付き数値計算に対して共有メモリ環境における実行時間の議論は多いが，大規模分散並列計算機上での実行時間と並列数の関係についての議論は多くない．そこで，本研究では，ScaLAPACK を利用して連立一次方程式に対する精度保証付き数値計算アルゴリズムを実装し，京コンピュータと FX100 上で実行時間を測定した．性能評価の結果，近似解を計算して，更にその誤差上限を精度保証付きで計算する場合の計算時間が，近似解の計算のみの場合の計算時間の 2 倍以下となることが明らかになった．

1. はじめに

現在，多くの計算機において IEEE 754 Standard[1] で定められた浮動小数点形式が利用されている．多くの CPU は 32bit 単精度または 64bit 倍精度の浮動小数点演算器を搭載しており，SIMD 化や FMA 演算器の搭載により，1 サイクルに実行可能な浮動小数点演算の回数は増加傾向にある．そのため，計算機を使用した計算では，浮動小数点形式を利用することが一般的となっている．

浮動小数点形式で表現可能な数は有限であり，実数に対して離散的に存在している．そのため，浮動小数点形式を利用した計算では切り上げや切り捨てといった処理（丸め）が発生する可能性があり，数学的に厳密な解に対して近似解が得られることになる．この近似解が厳密解に対してどの程度正しいのかを把握することは非常に重要であるが，その差を求めるのは一般的に容易ではない．そこで，数学的なアプローチにより近似解の正しさを検証する方法が研究されており，その計算法を精度保証付き数値計算と呼ぶ [2]．

精度保証付き数値計算では，計算された近似解に対して，浮動小数点数を用いて誤差評価を行い，その過程で生じるすべての誤差を考慮する．一方，近似解を計算する演算量に対して，近似解を評価をする演算量は一般的に多く，場合によっては高精度演算など演算量の多い計算も必要となる．そのため精度保証付き数値計算は，近似解を評価できるメリットはあるものの，実行時間の面で課題がある，という認識が一般的である．

一方，近年の計算機では，メモリアクセスコストが浮動小数点演算コストに対して相対的に大きいことや，SIMD 化の可否やマルチコアを活用できる並列性の有無なども，実行時間に大きく影響する．更に，大規模な分散並列計算においては，ネットワークを介した通信のコストが実行時間の対部分を占める場合も多い．そのため，もはや浮動小数点演算の回数のみで実行時間を議論することは現実的ではなく，対象とする計算機環境に応じて，様々な要因を考慮して議論する必要がある．

本研究では，科学技術計算に現れる線形計算の代表例の一つである，密行列を係数とする連立一次方程式の近似解の精度を保証する計算法を扱う．この計算法を共有メモリ環境における実行時間・精度・適応範囲などの観点から議論した先行研究として，密行列における LU 分解とその事前誤差解析を用いた高速な手法 [3]，倍精度の範囲において適応範囲が広い手法 [4]，悪条件問題にも有効な手法 [5]，問題に応じて計算量の少ないアルゴリズムに分岐を行う手法 [6]，高精度計算 [7] を用いて近似解とその誤差上限を改善する手法 [8] などが存在する．また，100CPU 程度の並列数における数万次元程度の計算に関しては，計算量を削減し高速化を行う方法 [9]，メモリ削減を行う方法 [10] などが提案されている．

しかし，文献 [9] では，使用された CPU はシングルコ

[1] 早稲田大学 基幹理工学部応用数理学科
[2] 理化学研究所 計算科学研究機構
[3] 北海道大学 情報基盤センター
[4] 明星大学 情報学部 情報学科
[a)] y.morikura@aoni.waseda.jp

アであり，アルゴリズム面に主眼が置かれたためか，並列数と実行時間の関係等はあまり報告されていない．また，京コンピュータのような最近の大規模分散並列環境における，実行時間と並列数の関係についての議論は著者らの知る限りでは報告されていない．そこで，本研究では，現在利用可能な ScaLAPACK のルーチンを利用して，連立一次方程式に対する精度保証付き数値計算アルゴリズムを実装し，京コンピュータ及び FX100 上で実行時間の評価を行った．性能評価の結果，並列数が一定以上のケースで，精度保証なしの場合（近似解の計算のみを行う場合）の実行時間に対して，精度保証を行う場合（近似解の計算を行い，更にその誤差上限も計算する場合）の実行時間が 2 倍以下となることを明らかにした．

以下，2 節では連立一次方程式に対する精度保証付き数値計算の概要について述べる．連立一次方程式の精度保証付き数値計算法には大きくわけて二つの方針（丸めモードの変更を用いるか用いないか）があり，本研究では丸めモードの変更を用いない事前誤差評価を用いた手法を用いる．これに関連し，浮動小数点演算における事前誤差評価，事前誤差評価を用いた連立一次方程式の精度保証付き数値計算における計算式についても 2 節で紹介する．3 節では，2 節で紹介した精度保証付き数値計算法について，ScaLAPACK を利用した実装方法を説明する．加えて，逐次計算の場合の浮動小数点演算のコストを述べ，精度保証がない場合とある場合の演算量を比較する．4 節では，京コンピュータと FX100 上で行った性能評価の結果について述べる．最後に 5 節で，本研究のまとめと今後の課題について述べる．

2. 連立一次方程式の近似解に対する精度保証付き数値計算

本節では，連立一次方程式の近似解の誤差を評価するための精度保証付き数値計算の概要を述べる．本研究では，密行列を係数とする連立一次方程式

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

を扱う．ただし，$\mathbb{F}$ は IEEE 754 Standard で定められた倍精度浮動小数点数の集合とし，$A \in \mathbb{F}^{n\times n}$, $b \in \mathbb{F}^n$ であるとする．つまり，問題の入力（係数行列と右辺ベクトル）には誤差は含まれていない，ということを意味している．一方，式 (1) に対して何らかの数値計算アルゴリズムを実行して得られた近似解を $\tilde{x}$ と表記する．

今回の目標は，式 (1) の真の（数学的に厳密な）解を x^* としたときに，計算で得られた近似解と未知である真の解との誤差上限

$$||x^* - \tilde{x}|| \leq \rho, \quad (\rho \in \mathbb{F}) \tag{2}$$

を浮動小数点演算を用いて求めることである．ただし，誤差上限 ρ の計算も浮動小数点を用いて行うので，その過程でも誤差が当然生じる可能性があるため，その誤差も考慮した上で，数学的に厳密な上限を求める．なお，本稿では，オーバーフローが起こらないと仮定する [*1]．

一般的に，数値計算で得られた連立一次方程式の近似解の精度を検証する場合，残差ベクトル

$$r := b - A\tilde{x} \tag{3}$$

を評価することが多い．しかし，もし，数学的に厳密な A の逆行列を計算することができれば，数学的には

$$A^{-1}r = A^{-1}(b - A\tilde{x}) = x^* - \tilde{x} \tag{4}$$

のように真の解と得られた近似解との誤差を求めることができる．しかし，実際の数値計算ではそもそも厳密な逆行列を得ることは不可能に近い．そこで，精度保証付き数値計算の分野では，式 (4) の代わりとして係数行列 A の近似逆行列を用いることができる以下の定理がよく用いられる．

定理 2.1 $\mathbb{R}$ を実数の集合とし, A, $R \in \mathbb{R}^{n\times n}$, $\tilde{x}$, $b \in \mathbb{R}^n$, I を単位行列とする．もし

$$||RA - I|| < 1 \tag{5}$$

を満たすならば，A は正則であり

$$||\tilde{x} - A^{-1}b|| \leq \frac{||R(A\tilde{x} - b)||}{1 - ||RA - I||} \tag{6}$$

を満たす．

なお，定理の中の演算は全て丸め誤差を含まない（数学的に厳密な）ものである．

上述の定理を利用することで，数値計算で計算可能な近似的な逆行列 R を用いて，残差から誤差の上限を計算するが可能である．ただし，定理の中の式 (5) 及び式 (6) は，実際には浮動小数点演算により計算される．そのため，数学的に厳密な誤差の上限を得るためには，これらの式の計算過程で生じる丸め誤差の影響も更に考慮する必要がある．

丸め誤差を考慮した計算方法の一例として大石，Rump は丸めモードの変更を用いた高速な手法 [3] を提案している．また，荻田，大石は 100CPU 程度の並列環境における丸めモードの変更を用いた手法 [9]，メモリ量を削減する手法 [10] を提案している．荻田らの手法 [9] は，大石，Rump の高速な手法 [3] に大規模問題用の事前誤差評価を導入したものである．荻田らの手法・環境では計算時間比は「近似解を求める時間：近似解を求める時間+精度保証」=「1：3」と高速であるが，LU 分解の事前誤差評価を用いているため精度保証が成功する適応範囲が狭い．また，メモリ量を削減する手法 [10] においては限られた計算機資源の環境

[*1] オーバーフローが起こるとそれ以降の計算に Inf が入る．そのため精度保証結果が Inf などになり結果として意味のないものになる．アンダーフローは起こってもよい．

においての考察を行っている．

しかし，これらの手法は丸めモードの変更を必要とする．丸めモードの変更は計算機環境によっては不可能，あるいは大きなオーバーヘッドを伴う場合がある．文献 [11] では CPU の丸めモード変更に要するコストが無視できないことを示している．

また，BLAS 等のライブラリを使用する場合も，ライブラリ内部の丸めモードの制御が不可能であったり，不明であったりする場合がある．そのため，丸めモードの変更に基づいた方法の適用が困難なケースが多々存在する．

そこで本研究では，丸めモードの変更が不可能な環境や，丸めモードの制御に対応しない BLAS[*2] を使うことを前提に，丸めモードの変更を必要としない方法を取り上げる．本研究では，IEEE754 Standard に従った最近点丸めのみを使用し，数学的な事前誤差評価と組み合わせた方法 [12], [13] を用いる．この方法では，事前誤差評価を用いて定理 2.1 を評価した以下の定理 2.2 を用いる．定理に用いられている事前誤差評価については付録を参照頂きたい．以下の定理では，ベクトル，行列のノルムを最大値ノルムとして計算している．$x \in \mathbb{R}^n$，$A \in \mathbb{R}^{m\times n}$ における最大値ノルムは

$$||x||_\infty := \max_{1\le i\le n} |x_i|, \quad ||A||_\infty := \max_{1\le i\le m} \sum_{j=1}^{n} |a_{ij}|$$

と定義する．$\mathbf{u}$ を相対精度（倍精度浮動小数点数では $\mathbf{u} = 2^{-53}$），$\mathbf{eta}$ をアンダーフローユニット（倍精度浮動小数点数では $\mathbf{eta} = 2^{-1074}$），$\mathbf{realmin} := \frac{1}{2}\mathbf{u}^{-1}\mathbf{eta}$ とする．$\mathrm{fl}(\cdots)$ は括弧内を最近点丸め（特に偶数丸め）を用いて計算することを意味する．このとき，括弧内の和と差の計算に関しては順不同である．例えば，$\mathrm{fl}(a+b+c)$ は $\mathrm{fl}(\mathrm{fl}(a+b)+c)$，$\mathrm{fl}(a+\mathrm{fl}(b+c))$ としてもよい．

定理 2.2 ([13]) $A \in \mathbb{F}^{n\times n}$, $b \in \mathbb{F}^n$ において，$\tilde{x} \in \mathbb{F}^n$ を数値計算で得られた近似解，I を単位行列，$R \in \mathbb{F}^{n\times n}$ を A の近似逆行列，$e = (1,1,\cdots,1)^T \in \mathbb{F}^n$ とする．ベクトル，行列の $|\cdot|$ は各要素ごとの絶対値を表すとすると

$$\begin{aligned}
&a_1 := \mathrm{muls}(|A|),\ a_2 := \mathrm{muld}(|R|,\ a_1),\\
&\alpha_1 := \mathrm{muls}(\mathrm{fl}(RA-I)),\ \alpha_2 := \mathrm{succ}(\mathrm{fl}((\mathrm{succ}(n)\mathbf{u})\cdot a_2)),\\
&\alpha_3 := \mathrm{fl}(\mathrm{succ}(n^2)\cdot\mathbf{eta}\cdot e)
\end{aligned}$$

$\max(\alpha_1) < 1$ であれば

$$\begin{aligned}
||RA-I||_\infty \le\ &||\mathrm{fl}(\mathrm{succ}(\alpha_1+\alpha_2+\alpha_3+\mathbf{u}e)\\
&+ 3\mathbf{u}\cdot\mathrm{ufp}(\alpha_1+\alpha_2+\alpha_3+\mathbf{u}e))||_\infty =: \alpha.
\end{aligned}$$

により $||RA-I||_\infty$ の上限が得られる．ただし $r \in \mathbb{F}$, $\mathrm{pred}(r) := \max\{f \in \mathbb{F} : f < r\}$, $\mathrm{succ}(r) := \min\{f \in \mathbb{F} : r < f\}$, $\mathrm{ufp}(r) := 2^{\lfloor \log_2 |r| \rfloor}$,

$$\begin{aligned}
\mathrm{muls}(A) &:= \mathrm{succ}(\mathrm{fl}(|A|e + ((n-1)\mathbf{u}\cdot\mathrm{ufp}(|A|e)))),\\
\mathrm{muld}(A,x) &:= \mathrm{succ}(\mathrm{fl}(|Ax| + ((n+2)\mathbf{u}\cdot\mathrm{ufp}(|A||x|)\\
&\qquad +\mathbf{realmin}\cdot e))).
\end{aligned}$$

また，

$$\begin{aligned}
\mathrm{mid} &= \mathrm{fl}(A\tilde{x}-b),\\
\mathrm{rad} &= \mathrm{fl}((n+3)\mathbf{u}\cdot\mathrm{ufp}(|A||\tilde{x}|+|b|) + \mathbf{realmin}\cdot e)
\end{aligned}$$

で残差の包含が得られ，

$$\begin{aligned}
&|R(A\tilde{x}-b)| \le (|R\mathrm{mid}| + |R|\mathrm{rad}),\\
&b_1 := \mathrm{muld}(R,\mathrm{mid}),\ b_2 := \mathrm{muld}(|R|,\mathrm{rad}),\\
&||R(A\tilde{x}-b)||_\infty \le \max(\mathrm{succ}(b_1+b_2)) =: \beta,
\end{aligned}$$

により $||R(A\tilde{x}-b)||_\infty$ の上限が得られる．以上，$||RA-I||_\infty$，$||R(A\tilde{x}-b)||_\infty$ の上限より真の解と近似解との誤差上限は

$$\begin{aligned}
||x^*-\tilde{x}||_\infty &\le \frac{\beta}{1-\alpha} \le \frac{\beta}{\mathrm{pred}(\mathrm{fl}(1-\alpha))}\\
&\le \mathrm{fl}\left(\mathrm{succ}\left(\frac{\beta}{\mathrm{pred}(1-\alpha)}\right)\right) =: err
\end{aligned}$$

と計算できる．

本定理における計算は全て $\mathrm{fl}(\cdots)$ の形で記載されているため，本定理をそのまま計算することで連立一次方程式の精度保証付き数値計算が可能である．ただし，影響する丸め誤差を過大に評価しているため，得られる誤差上限は丸めモードの変更を用いて定理 2.1 を計算したものよりも精度が悪い．

3. ScaLAPACK に基づく実装と分析

現状，分散並列環境において密行列計算を行う場合，ScaLAPACK を利用するのが容易かつ一般的な方法であると言える．そこで，今回は研究の第一段階として，ScaLAPACK を用いた実装を性能評価の対象とする．本節では，ScaLAPACK に基づく実装の概要を述べ，各計算ステップについて簡単に分析を行う．

まず，近似解の計算であるが，係数行列が密行の場合，LU 分解を行い，前進・後退代入により近似解を求める方法が一般的である．ScaLAPACK で提供されているルーチンとしては，pdgetrf（LU 分解）と pdgetrs（前進/後退代入）が該当する．

一方，近似解の誤差上限の計算では，最初に近似逆行列 R を陽的に生成する必要がある．今回は近似解の計算において LU 分解が行われていることを前提とし，LU 分解の結果（L 及び U ファクター）から近似逆行列を構成するとする．これは，ScaLAPACK の pdgetri ルーチンにより行

[*2] 精度保証において行列積ライブラリにはストラッセンのアルゴリズムなど行列積の計算量を削減するアルゴリズムを使われていないことが必要である．

表 1 連立一次方程式の求解と近似解の誤差上限に関する精度保証付き数値計算のステップと演算量（逐次計算の場合）

	計算内容	ルーチン名	演算量
(1)	LU 分解	pdgetrf	$\frac{2}{3}n^3$
(2)	前進・後退代入	pdgetrs	$O(n^2)$
(3)	近似逆行列の生成	pdgetri	$\frac{4}{3}n^3$
(4)	RA の計算	pdgemm	$2n^3$
(5)	$R((A\tilde{x})-b)$ の計算	pdgemv	$O(n^2)$
	近似解の計算のみ（1 と 2）		$\frac{2}{3}n^3+O(n^2)$
	近似解と誤差上限の計算（1 から 5）		$4n^3+O(n^2)$

う．次に，PBLAS の pdgemm ルーチンを用いて，行列積 RA を計算する．最後に，PBLAS の pdgemv ルーチンにより，行列ベクトル積を 2 回行い，$R((A\tilde{x})-b)$ を計算する．

上述の計算内容を逐次計算の場合の演算量とともに**表 1** にまとめる．表 1 から分かるように，近似解のみを求める場合に対して，近似解の計算に加えて誤差上限を計算する場合の逐次計算における演算量は約 6 倍となる．そのため，演算量だけで両者を比較した場合，精度保証付き数値計算はコスト面において計算時間の観点で高コストだと言える．

しかし，分散並列計算を想定した場合，LU 分解，逆行列の計算，行列積は，それぞれの処理内容から強スケーリングの振る舞いが異なることが予想される．例えば，行列積の計算は，演算律速であり並列性が高く，処理間の依存関係も少ないため，通信と計算のオーバーラップ等も可能である．更に，各ベンダーが高性能実装に力を入れていることが多いという現実的な点もあり，プロセス数を増やしても良くスケールすることが一般的に期待できる．逆に，LU 分解の計算は，恐らく，上記の 3 種類の処理の中では最も逐次性が強く，並列処理の粒度も細かい．そのため，プロセス数を増やした場合しても，通信（や同期）のコストにより，実行時間の減少には限界があると考えられる．

以上の点を踏まえると，逐次計算の場合の演算量が 6 倍だからといって，必ずしも並列計算，特にプロセス数が十分に多い場合において，精度保証がない場合（近似解のみを求める）に対して，精度保証を行う場合（近似解の計算に加えて，その評価を精度保証付き数値計算で行う）の実行時間が 6 倍程度となるとは限らない．そこで，今回，精度保証がない場合と精度保証がある場合の実行時間の比較を，実際に京コンピュータと FX100 を用いて行う．

4. 性能評価

本節では，京コンピュータ，FX100 上で ScaLAPACK を用いた連立一次方程式の精度保証付き数値計算の性能を評価した結果を示す．

今回，性能評価に使用した京コンピュータと FX100（HOKUSAI-GreatWave）の主な仕様は**表 2** の通りである．

表 2 性能評価環境の主な仕様

項目	京コンピュータ	FX100
CPU 数/ノード	1	1
プロセッサ	SPARC64 VIIIfx 8 コア 2.0GHz	SPARC64 XI fx 32 コア 1.975GHz
ピーク演算性能/ノード	128GFLOPS	1TFLOPS
メモリ総量/ノード	16GB	32GB
メモリバンド幅/ノード	64GB/s	240GB/s （read/write）
ネットワーク	Tofu 5 GB/s （双方向）	Tofu 2 12.5 GB/s （双方向）

表 3 性能評価の条件

項目	京コンピュータ	FX100
コンパイラ	mpifccpx	mpifccpx
コンパイル オプション	-Xg -Kfast -Kparallel -Kopenmp -O0 -MD -SCALAPACK -SSL2BLAMP -DKCOMPUTER	-Xg -Kfast Kparallel -Kopenmp -O0 -MD -SCALAPACK -SSL2BLAMP -DFX100
MPI/OpenMP の実行形態	1 プロセス/ノード 8 スレッド/プロセス	2 プロセス/ノード （1 プロセス/CMG） 16 スレッド/プロセス

また，プログラムは C 言語で作成し，それぞれの環境でベンダーにより提供されている ScaLAPACK のライブラリをリンクした．コンパイル時のオプションと実行時の MPI/OpenMP のハイブリッド実行形態は**表 3** の通りである．なお，FX100 では，1 プロセッサあたり 2 つのコアメモリグループ（CMG）を持っているので，今回は CMG ごとに MPI プロセスを 1 つ割り当てる形とした．また，どちらの環境でも，ScaLAPACK に関しては，プロセスを 2 次元のグリッド（形状は正方形もしくはそれに近い形）に割り当て，行列データは，ブロック幅を 64 として二次元ブロックサイクリック分散で保持するように設定した．係数行列 A は乱数行列を用いた．また，それぞれの真の解のベクトルの各要素が 1 に近くなるように，右辺ベクトル b は $\mathrm{fl}(A\cdot e)$ のように作成した（e は各成分が 1 のベクトル）．

まず，行列サイズを $n=10240$ として，プロセス数を変えて，精度保証なし（近似解の求解のみ）の場合の実行時間と精度保証あり（近似解の求解とその精度を精度保証付き数値計算で評価）の場合の実行時間を測定した結果を，両者の比とともに**表 4**（京コンピュータ），**表 5**（FX100）に示す．二つの表より，京コンピュータ上では 8 プロセス

表 4 精度保証あり/なしの場合の実行時間と両者の比（京コンピュータ, $n = 10240$）

プロセス数	実行時間（秒）		実行時間の比
	精度保証なし	精度保証あり	（あり/なし）
1	18.525	64.770	3.496
2	23.174	55.040	2.375
4	11.817	26.100	2.209
8	10.738	18.104	1.686
16	5.080	9.544	1.879
32	2.859	5.596	1.957
64	2.232	3.988	1.787
128	2.523	4.306	1.707
256	2.205	3.130	1.420
512	2.573	3.874	1.506
1024	2.410	3.104	1.288

を，FX100 上では 2 プロセスを超えた段階で，両者の実行時間の比が 2 以下となることが確認できる．

次に，精度保証ありの場合と精度保証なしの場合の実行時間の比について，行列サイズを変えてプロセス数との関係を調査した結果を**図 4** に示す．このグラフより，以下のことが読み取れる．

- 4 つの行列サイズのいずれについても，プロセス数を十分多くすると，実行時間の比が 2 以下となる．
- 同じプロセス数では，行列サイズが大きくなるほど，実行時間の比が大きくなる傾向がある．
- 同じ行列サイズで同じプロセス数の場合，FX100 の方が京コンピュータより実行時間の比が小さい．

これらの結果をもう少し詳しく検証するために，表 1 に示した主要な計算カーネルごとに ScaLAPACK のルーチンの性能を評価した．行列サイズを 10240 としてプロセス数を変えて各ルーチンの実行時間を測定した結果を**図 2**（京コンピュータ）と**図 3**（FX100）に示す．二つのグラフより，今回の二つの環境で提供されている ScaLAPACK に関しては，pdgemm（行列積），pdgetri（LU 分解の結果から逆行列の生成）, pdgetrf（LU 分解）の順に強スケーリングが悪くなっていることが確認できる．特に，pdgemm と他の二つのルーチンとの間には強スケーリングに関して非常に大きな差があることが分かる．また，pdgemv（行列ベクトル積）はプロセス数を増やしてもほとんど実行時間が減少しておらず，その結果として，例えば京の 1024 プロセスの場合には，pdgemm よりも pdgemv の方が実行時間が大きくなってしまっていることが示されている．

以上の結果をまとめると，連立一次方程式の近似解を求めるためのルーチンと精度保証付き数値計算で誤差を評価する際に使用するルーチンとの間には強スケーリングに関して違いがあり，後者の方がよりスケールするために，プロセス数を増やすと後者の実行時間が相対的に少なくなり，精度保証ありと精度保証なしの場合の実行時間の比が小さくなったと判断できる．

表 5 精度保証あり/なしの場合の実行時間と両者の比（FX100, $n = 10240$）

プロセス数	実行時間（秒）		実行時間の比
	精度保証なし	精度保証あり	（あり/なし）
1	8.307	21.665	2.608
2	14.9451	23.186	1.551
4	7.7437	12.037	1.554
8	5.0395	7.493	1.487
16	2.7523	7.493	1.654
32	2.6963	3.956	1.467
64	2.2335	3.142	1.407

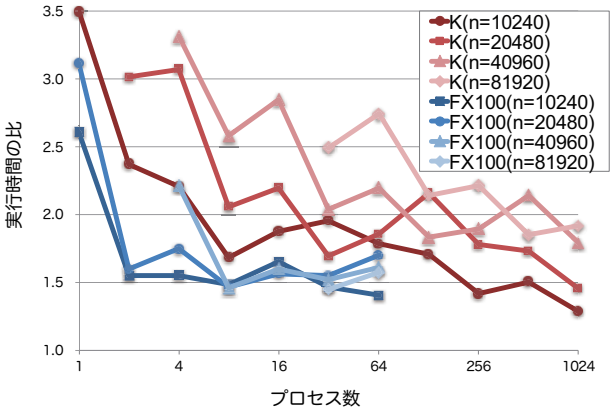

図 1 実行時間の比（精度保証あり/なし）（京コンピュータ及び FX100）

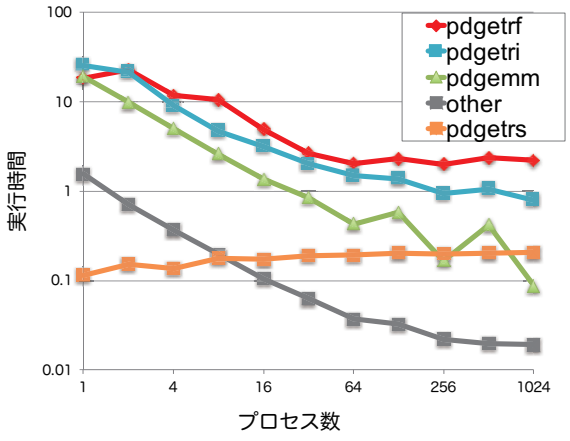

図 2 ScaLAPACK の各ルーチンの実行時間（京コンピュータ, $n = 10240$）

最後に，実際に計算された近似解の誤差上限の一例として，京コンピュータで 1024 プロセスを用いた場合の結果を**表 6** に示す．今回の性能評価では，（右辺ベクトルの生成時の誤差を無視すれば）解ベクトルの各要素が 1 となるように設定しているため，表 6 の結果は，例えば，行列サイズが 10240 の場合には近似解ベクトルの各要素は 10 進

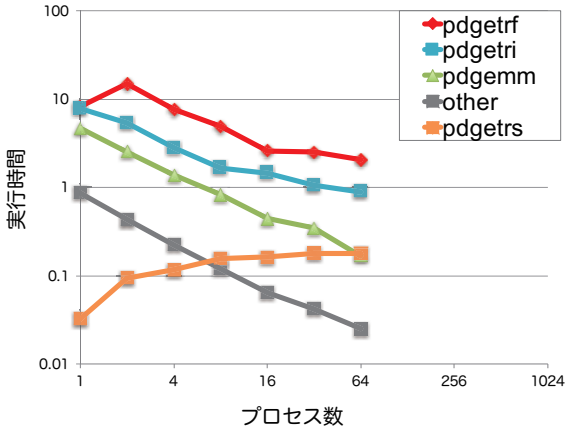

図 3 ScaLAPACK の各ルーチンの実行時間
（FX100, $n = 10240$）

表 6 計算された近似解の誤差 $||x^* - \tilde{x}||_\infty$ の上限
（京コンピュータ, 1024 ノード）

行列サイズ	$||x^* - \tilde{x}||_\infty$ の上限
10240	2.20E-05
20480	1.51E-04
40960	1.15E-04
81920	3.12E-03

で約 5 桁は正しいことが保証されている，ということを意味している．より誤差上限をシャープに計算する方法とその性能評価などについては今後の課題である．

5. まとめと今後の課題

従来，精度保証付き数値計算は，主に浮動小数点演算回数が多いという理由から，計算時間の観点で高コストがあると認識されてきた．しかし，昨今の計算機環境では，浮動小数点演算の回数だけでなく，様々な要因が実行時間に影響することが知られており，使用する計算機環境の特徴を考慮した議論が必要である．本研究では，大規模分散並列環境において，密行列を係数とする連立一次方程式の近似解の精度を精度穂所付き数値計算で評価する場合の実行時間の検証を目的とし，研究の第一段階として，ScaLAPACK に基づく実装について，京コンピュータと FX100 上で性能評価を行った．

今回の性能評価を通して，近似解を計算する処理（LU 分解）と近似解に対して精度保証付き数値計算を行う際の処理（近似逆行列の計算や行列積の計算）は強スケーリングの挙動が異なり，京コンピュータや FX100 で提供されている ScaLAPACK では，後者の方がプロセス数を増やしたときの実行時間の減少が大きいことが確認された．その結果，1 プロセスの場合では，精度保証なしの場合に対して，精度保証ありの場合の実行時間が 3 倍前後であるが，プロセス数を増やすことで実行時間の比が小さくなり，一定数のプロセス以上を用いる場合には，両者の比が 2 以下となることが確認できた．

今後の課題としては，まず，今回は各環境でベンダー提供の ScaLAPACK を用いた実装であり，その最適化の状況等を議論する必要がある．例えば，LU 分解については，HPL のベンチマーク等の関係で，高度に最適化された実装が ScaLAPACK とは別に存在する可能性も高い．同時に，実機上での実行時間の測定だけではなく，モデル等を用いて演算・通信のコストを評価し，各処理のスケーリングを定性的に議論する必要もある．

また，今回は ScaLAPACK を利用するということで，精度保証の際の処理において，ScaLAPACK のルーチンを順番にコールするという形をとったが，最初から精度保証まで行うことが分かっているのであれば，複数の処理を組み合わせて行うことで通信時間を削減する，といったアプローチが可能となる可能性もある．また，分散並列環境での各ルーチンの実行時間の挙動を踏まえて，精度保証付き数値計算の内部で行う処理をより都合のよいものに変更する（例：LU 分解ではなく QR 分解で近似解を計算する [14] ということも検討する価値がある）．加えて，実行時間は許容範囲だったとしても，そもそも計算された誤差の上限（例えば表 6 に示した数値）が許容できない場合も十分に有り得る．この場合，今回の方法では，事前誤差評価により丸め誤差を評価している点と，そもそもの近似解の精度が悪いという点の二つの可能性が考えられる．前者に対する対応策としては，高精度演算を部分的に組み合わせることで，近似解を改善したり，誤差評価の精度を改善したりする方法が研究されており，結果として，近似解の精度を 10 桁以上保証できる場合 [8] があることなどが報告されている．このような手法について，実行時間を詳しく評価することも今後の重要な課題の一つである．

謝辞 本研究に関して，多くの有益なご助言を頂戴致しました理化学研究所計算科学研究機構の今村俊幸チームリーダーに深く感謝いたします．

本研究は JSPS 科研費「15K15939」,「15H02709」の助成を受けた．本研究における京コンピュータで得られた結果は理化学研究所 計算科学研究機構（課題 ID：ra000005），理化学研究所 情報基盤センター HOKUSAI GreatWave システムによって得られたものである．

付　　録

IEEE 754 Standard に従った浮動小数点演算においては四則演算は

$$\mathrm{fl}(a \circ b) = (a \circ b)(1+\epsilon),\ |\epsilon| \le \mathbf{u},\ \circ \in \{+,-\},$$
$$\mathrm{fl}(a \circ b) = (a \circ b)(1+\epsilon) + \eta,$$
$$|\epsilon| \le \mathbf{u},\ |\eta| \le \mathbf{eta}/2,\ \epsilon\eta = 0,\ \circ \in \{\times, /\}$$

を満たすため，一回の浮動小数点演算に入る丸め誤差の上

限を見積もることができる．

四則演算の拡張により浮動小数点演算を用いた総和と内積の誤差評価が提案されている [15], [16]．まず Rump が考案した ufp (unit in the first place) と呼ばれる浮動小数点数の最初の bit の情報について定義 [16] する．ufp は以下のように定義される：

$$0 \neq r \in \mathbb{R} \Rightarrow \mathrm{ufp}(r) := 2^{\lfloor \log_2 |r| \rfloor}$$

このとき，$\mathrm{ufp}(0) = 0$ とする．例えば，$\mathrm{ufp}(0.6) = 0.5$, $\mathrm{ufp}(1.1) = 1$ のように計算され，$\mathrm{ufp}(\cdot)$ の計算は浮動小数点演算を用いて 4flops で計算することができる．また，ufp の引数が行列・ベクトルの場合は成分毎に ufp の計算を行うとする．

このとき $x \in \mathbb{F}^n$ の総和の誤差評価は

$$\left| \mathrm{fl}\left(\sum_{i=1}^{n} x_i\right) - \sum_{i=1}^{n} x_i \right| \leq (n-1)\mathbf{u} \cdot \sum_{i=1}^{n} |x_i|,$$
$$\left| \mathrm{fl}\left(\sum_{i=1}^{n} x_i\right) - \sum_{i=1}^{n} x_i \right| \leq \mathrm{fl}\left((n-1)\mathbf{u} \cdot \mathrm{ufp}\left(\sum_{i=1}^{n} |x_i|\right)\right).$$

$|x| = (|x_1|, |x_2|, \cdots, |x_n|)$ としたとき，$x, y \in \mathbb{F}^n$ における内積の誤差評価は，$n\mathbf{u} < 1$ のとき

$$|\mathrm{fl}(x^T y) - x^T y| \leq \delta_n |x|^T |y| + (2n-1) \cdot \mathbf{eta}/2.$$

ここで $\delta_n := (n\mathbf{u} + n\mathbf{u^2})$[*3]．また，$2(n+2)\mathbf{u} < 1$ のとき

$$|\mathrm{fl}(x^T y) - x^T y| \leq \mathrm{fl}((n+2)\mathbf{u} \cdot \mathrm{ufp}(|x|^T |y|) + \mathbf{realmin}),$$

ここで，$\mathbf{realmin} := \frac{1}{2}\mathbf{u}^{-1}\mathbf{eta}$ とし，**realmin** は浮動小数点数の正規化数の中で最も小さい正の数を意味する．

総和，内積の事前誤差評価において ufp を用いたものは $\mathrm{fl}(\cdots)$ で評価されているためそのまま，計算機に実装できる．浮動小数点演算の事前誤差評価の詳細については文献 [15], [16], [17] などを参照頂きたい．

次に，一回の四則演算における包含を行うため，ある浮動小数点数におけるマイナス方向へ一つずらした浮動小数点数（$\mathrm{pred}(\cdot)$），プラス方向へ一つずらした浮動小数点数（$\mathrm{succ}(\cdot)$）を定義する [18]．$r \in \mathbb{F}$ において

$$\mathrm{pred}(r) := \max\{f \in \mathbb{F} : f < r\},$$
$$\mathrm{succ}(r) := \min\{f \in \mathbb{F} : r < f\}.$$

ある浮動小数点数おマイナス方向へ一つずらした浮動小数点数，プラス方向へ一つずらした浮動小数点数も最近点丸めを用いて求めることができる，実装方法は文献 [18] を参照頂きたい．よって二つの浮動小数点数の演算 $a, b \in \mathbb{F}$ $\circ \in \{+, -, \cdot, /\}$ において

$$\mathrm{pred}(\mathrm{fl}(a \circ b)) < a \circ b < \mathrm{succ}(\mathrm{fl}(a \circ b))$$

が成り立つ．

[*3] 近年 δ_n を $n\mathbf{u}$ で置き換えた誤差評価式が提案されている [17]

参考文献

[1] ANSI/IEEE 754–2008: IEEE Standard for Floating-Point Arithmetic, IEEE, New York, 2008.
[2] 大石 進一，現代非線形科学シリーズ 6 精度保証付き数値計算（コロナ社，2001 年）
[3] S. Oishi and S.M. Rump, Fast verification of solutions of matrix equations, Numer. Math., 90(4):755–773, 2002.
[4] Atsushi Minamihata, Kouta Sekine, Takeshi Ogita, Siegfried M. Rump and Shin'ichi Oishi, Improved error bounds for linear systems with H-matrices, Nonlinear Theory and Its Applications, IEICE, Vol. 6, No. 3 pp. 377-382, 2015.
[5] 太田 貴久，荻田 武史，S. M. Rump，大石 進一，悪条件連立一次方程式の精度保証付き数値計算法，日本応用数理学会論文誌，15:3，269–287，2005.
[6] Katsuhisa Ozaki, Takeshi Ogita and Shin 'ichi Oishi, An algorithm for automatically selecting a suitable verification method for linear systems, Numerical Algorithms, 56:3, pp. 363–382, 2011.
[7] T. Ogita, S.M. Rump, and S. Oishi. Accurate sum and dot product. SIAM Journal on Scientific Computing (SISC), 26(6):1955–1988, 2005.
[8] 大石 進一，荻田 武史，太田 貴久，高精度内積計算アルゴリズムを用いた連立一次方程式の精度保証付き数値計算法，シミュレーション 25(3)，170–178，2006.
[9] 荻田 武史，大石 進一，大規模連立一次方程式のための高速精度保証法，情報処理学会論文誌 数理モデル化と応用 46:SIC10(TOM12), 10–18, 2005.
[10] 荻田 武史，大石 進一，連立一次方程式のメモリ量を低減した精度保証付き数値計算法，シミュレーション 25(3)，179–184，2006.
[11] S. Rump, T. Ogita, Y. Morikura, and S. Oishi. Interval arithmetic with fixed rounding mode. Nonlinear Theory and its Applications (IEICE), 7(3):362–373, 2016.
[12] T. Ogita, S.M. Rump, and S. Oishi, Verified solution of linear systems without directed rounding, Technical Report 2005-04, Advanced Research Institute for Science and Engineering, Waseda University, Tokyo, Japan, 2005.
[13] Yusuke Morikura, Katsuhisa Ozaki and Shin 'ichi Oishi, Verification methods for linear systems using ufp estimation with rounding-to-nearest, Nonlinear Theory and its Applications, IEICE, vol.4, no.1, pp. 12-22, 2013.
[14] 柳澤 優香，大石 進一，野田 ふみ，ハウスホルダー QR 分解を用いた連立一次方程式の数値解に対する精度保証，日本応用数理学会 2016 年度年会 講演予稿集，2016.
[15] N.J. Higham, Accuracy and Stability of Numerical Algorithms, second edition, SIAM Publications, Philadelphia, 2002.
[16] S.M. Rump. Error estimation of floating-point summation and dot product. BIT Numerical Mathematics , 52(1):201–220, 2012.
[17] C.-P. Jeannerod and S.M. Rump. Improved error bounds for inner products in floating-point artihmetic. SIAM. J. Matrix Anal. & Appl. (SIMAX), 34(2):338–344, 2013.
[18] S.M. Rump, P. Zimmermann, S. Boldo, and G. Melquiond. Computing predecessor and successor in rounding to nearest. BIT Numerical Mathematics, 49(2):419–431, 2009.