任意精度保証計算のための多倍長演算における仮数部のビット長の適応制御の提案と評価

中島大雅[†1]　古賀雅伸[†1]　矢野健太郎[†1,*1]

本論文では，多倍長演算における仮数部のビット長を適応的に調節することで任意の結果精度を保証できる手法を提案する．そして，提案手法に基づき数値計算パッケージを開発し，具体的な問題に適用し評価する．

Adaptive Bit-Length Control of Mantissa in Multiple-Precision Arithmetic for Numerical Computation with Guaranteed Arbitrary Accuracy

Hiromasa Nakashima ,[†1] Masanobu Koga [†1]
and Kentaro Yano[†1,*1]

This paper proposes a method for numerical computations with guaranteed arbitrary accuracy by adaptively controlling the bit-length of the mantissa in multiple-precision arithmetic. We have developed a numerical computation package based on the proposed method and evaluated the method by applying it to some numerical computation problems.

1. はじめに

一般に数値計算では倍精度の浮動小数点型が用いられる．数学的に厳密な式が与えられたとしても，丸め誤差などが混入するため，計算機上では精度の高い結果を得られるとは限らない．

†1 九州工業大学
Kyushu Institute of Technology
∗1 現在，福岡工業大学短期大学部
Presently with Fukuoka Institute of Technology Junior College

表 1　演算精度と結果精度
Table 1　Precision and accuracy

計算方法	演算精度	結果精度
単・倍精度	低い（不変）	判別不可
精度保証付き倍精度	低い（不変）	判別可
多倍長	高い（可変）	判別不可
精度保証付き多倍長	高い（可変）	判別可

数値計算における精度には，演算精度（precision）と結果精度（accuracy）の 2 つの精度がある[1)]．演算精度とは，浮動小数点数の仮数部の桁数 p であり，IEEE754 規格では単精度の場合 $p = 24$，倍精度の場合 $p = 53$ である．これに対し，結果精度は，真値に対する計算結果の誤差であり，問題に依存するため演算精度からだけでは分からない．また，計算結果を見ても，その解の結果精度がどのくらいあるかを判断することは難しい．

精度桁を任意に設定できる多倍長演算を用いれば，演算精度を高くすることはできるが，結果精度を知ることは難しい．一方，精度保証付き数値計算[2)] を用いれば，誤差の大きさを見積もることができ，結果精度を知ることはできるが，その結果精度が低すぎた場合，その値を採用することはできない．

近年，精度保証付き数値計算と多倍長演算を組み合わせた精度保証付き多倍長計算が報告されている[3),4)]．この精度保証付き多倍長計算を使うと，演算精度を高くすることができ，その結果精度を知ることができる．これらの計算方法と精度の関係を**表 1** に示す．

しかし，与えられた（指定された）結果精度を達成するには，演算精度がどのくらい必要なのかを事前に知ることができないため，多倍長演算での演算精度をトライ・アンド・エラーにより設定することになる．

本論文では，多倍長演算における仮数部のビット長を適応的に調節することで任意の結果精度を保証できる手法を提案する．そして，提案手法に基づき数値計算パッケージを開発し，具体的な問題[5)] に適用し評価する．

2. 結果精度を指定可能な計算手法

2.1 演算精度と失われる精度の桁数の関係

ここでは，仮数部のビット長と数値計算誤差によって失われる精度の桁数の関係を予備実験によって調べる．予備実験には，3 章で述べる多倍長演算パッケージ MPFloat と精度保

表 2 多倍長の精度桁と失われた精度桁数（$n = 10$）
Table 2 Used precision and lost accuracy ($n = 10$)

多倍長の仮数部	失われた最大の精度桁数（2 進数）
128bit	**43bit**
196bit	**43bit**
256bit	**43bit**
320bit	**43bit**
384bit	**43bit**

表 3 多倍長の精度桁と失われた精度桁数（$n = 20$）
Table 3 Used precision and lost accuracy ($n = 20$)

多倍長の仮数部	失われた最大の精度桁数（2 進数）
128bit	**93bit**
196bit	**93bit**
256bit	**93bit**
320bit	**93bit**
384bit	**93bit**

証付き数値計算パッケージ CGA を用いた．

2.1.1 線形方程式

次の線形方程式の解の導出問題について考える．

$$Ax = b$$

ただし，A は n 次のヒルベルト行列，b はすべての要素が 1 の n 次のベクトルである．^{T}

この線形方程式（$n = 10, n = 20$）を精度保証付き多倍長演算[6] で複数の精度桁について解き，それぞれの失われた精度の桁数を求めた．その結果を**表 2** と**表 3** に示す．ただし，係数行列はそれぞれの桁数で生成した．また，失われた精度桁数を，

$$\text{失われた精度桁数} = \text{多倍長の精度桁数} - \left\lfloor \log_2 \left| \frac{\mathrm{r}}{\mathrm{c}} \right| \right\rfloor$$

とする．ただし，c と r は精度保証付き数値計算で求まった値の中心と半径であり，$\lfloor \cdot \rfloor$ は下向きに丸めた整数を表す．なお，得られた解の各成分について失われた精度桁数を求め，これらの最大値が表に示されている．この結果より，線形方程式の解の導出問題の場合，演算に用いる仮数部のビット長に関係なく，同じ精度の桁数が失われることが分かる．

2.1.2 極配置問題

次のシステムの極配置問題[5] について考える．

$$\dot{x} = Ax + Bu$$

ただし，

表 4 多倍長の精度桁と失われた精度桁数（$e = 10^5$）
Table 4 Used precision and lost accuracy ($e = 10^5$)

多倍長の仮数部	失われた最大の精度桁数（2 進数）
128bit	**18bit**
196bit	**18bit**
256bit	**18bit**
320bit	**18bit**
384bit	**18bit**

表 5 多倍長の精度桁と失われた精度桁数（$e = 10^{10}$）
Table 5 Used precision and lost accuracy ($e = 10^{10}$)

多倍長の仮数部	失われた最大の精度桁数（2 進数）
128bit	**35bit**
196bit	**35bit**
256bit	**35bit**
320bit	**34bit**
384bit	**34bit**

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \times e \end{bmatrix}, \; B = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T$$

である．状態フィードバック，

$$u = -Fx$$

により，閉ループ系 $\dot{x} = (A - BF)x$ の極（$A - BF$ の固有値）が $\{-1, -5, -10, -15, -20, -25\}$ となるよう 1 行 6 列のゲイン行列 F を求める．上に示す極配置問題（$e = 10^5, e = 10^{10}$）を精度保証付き多倍長演算（文献 5) のアルゴリズムに区間行列演算[2),7)] を適用）で複数の精度桁について解き，それぞれの失われた精度桁数を求めた．その結果を**表 4** と**表 5** に示す．

この結果より，極配置問題の場合，演算に用いる仮数部のビット長に関係なく，ほとんど同じ精度の桁数が失われることが分かる．

他の問題についても調べたところ，同様の結果を得た．このことから，精度保証付き多倍長演算において，失われる精度の桁数は問題に依存するが，演算に用いる仮数部のビット長にほとんど関係ないと考えられる．

2.2 指定された結果精度を保証する手法

ここでは，前節で得られた予備実験の結果に基づき，指定された結果精度が保証される解を得るための手法を提案する．基本的なアイディアは，適当な仮数部のビット長で精度保証付き多倍長演算により解を求め，得られた解が保証する結果精度が，指定された精度を満た

すか判定し，満たしていない場合，適当に仮数部のビット長を増やし，再計算を行うという処理を繰り返すというものである．精度を絶対誤差で指定する手法と精度桁で指定する手法を以下に示す．

2.1 節の結果より，仮数部のビット長に関係なくほとんど同じ精度桁数が失われると考えられるので，多くの場合に 1 回目の演算で指定された精度を満たすために必要な仮数部のビット長を求めることができ，2 回目の演算で指定された精度（誤差）を保証した解を得ることができる．

2.3 精度桁指定

p^* を指定された精度の桁数（ビット数）とする．k 回目の精度保証付き数値計算に用いる仮数部ビット数を p_k とする．初期ビット数を，

$$p_0 = p^* \tag{1}$$

とする．k 回目の，精度保証付き数値計算で求まった値の中心 c_k と半径 r_k から，その値の有効桁数（ビット数），

$$\hat{p}_k = \log_2 |c_k/r_k| \tag{2}$$

を求める．ただし，求まった値が行列の場合，成分ごとに計算した有効桁数（ビット数）の最大値を $\hat{p}_k$ とする．そして，指定された精度の桁数 p^* と $\hat{p}_k$ の差をとることで増加させるビット数，

$$\Delta p_k = \lceil p^* - \hat{p}_k \rceil \tag{3}$$

を求める．ただし $\lceil \cdot \rceil$ は上向きに丸めた整数を表す．使用したビット数 p_k に，指定された精度 p^* を満たすために足りないビット数 Δp_k を加えることで次に使用するビット数，

$$p_{k+1} = p_k + \Delta p_k \tag{4}$$

を求める．

なお，仮数部のビット長の変更単位が N の場合（Exflib[8] の場合は，64bit）は，式 (3) を，

$$\Delta p_k = \lceil (p^* - \hat{p}_k) \times a/N \rceil \times N \tag{5}$$

とする．

2.4 絶対誤差指定

e^* を指定された誤差，k 回目の精度保証付き数値計算に用いる仮数部のビット数を p_k とする．

精度が絶対誤差で指定された場合，どの位のビット数を演算に用いればよいかわからないので，初期ビット数を Exflib で設定可能な最小の仮数部のビット数，

$$p_0 = 128 \tag{6}$$

とする．k 回目の精度保証で求まった値の中心 c_k と半径 r_k から，その値の有効桁数（ビット数），

$$\hat{p}_k = \log_2 |c_k/r_k| \tag{7}$$

を求める．ただし，求まった値が行列の場合，成分ごとに計算した有効桁数（ビット数）の最大値を $\hat{p}_k$ とする．そして，指定された絶対誤差 e^* を保証するために必要なビット数，

$$p_k^* = \log_2 |c_k/e^*| \tag{8}$$

を求める．指定されたビット数 p_k^* と $\hat{p}_k$ の差をとることで増加させるビット数，

$$\Delta p_k = \lceil p_k^* - \hat{p}_k \rceil \tag{9}$$

を求める．使用したビット数 p_k に指定された誤差を保証するために足りないビット数 Δp_k を加えることで次に使用するビット数，

$$p_{k+1} = p_k + \Delta p_k \tag{10}$$

を求める．

なお，仮数部のビット長の変更単位が N の場合（Exflib の場合は，64bit）は，式 (9) を，

$$\Delta p_k = \lceil (p_k^* - \hat{p}_k) \times a/N \rceil \times N \tag{11}$$

とする．

3. 数値計算環境

近年，GMP（C）[9]，Exflib（C++）[8]，BigDecimal（Java），などの多倍長精度計算環境を提供するライブラリが開発され，有効な適用事例が報告されている．

我々が開発している数値計算環境を**図 1** に示す．基盤数値計算パッケージ NFC（Numerical FoundationClasses）[10] は倍精度などのプリミティブなデータ型および汎用的な数値データ型に基づく基本演算機能（LU 分解，QR 分解などの行列演算や複素数演算など）を提供する．多倍長演算パッケージ MPFloat[11] は，C++で実装された多倍長演算パッケージ Exflib，及び GMP（MPFR[12]）を Java から利用するためのパッケージであり，NFC の汎用データ型に対応しているので，NFC と MPFloat を組み合わせて利用することで多倍長精度での LU 分解，QR 分解などの行列演算や複素数演算を行うことができる． 精度保証付き数値計算パッケージ CGA（Computing with Guranteed Accuracy）[13] は，倍精度での精度保証付き数値計算及び MPFloat を用いた多倍長精度での精度保証付き数値計算を行うことができる．本研究で開発する任意精度保証計算パッケージ GAP（Guranteed à Priori Precision）では，CGA と MPFloat を用いることで指定された結果精度を保証する演算を実現する．

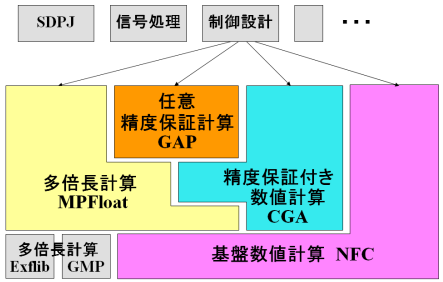


図 1 数値計算環境の構造
Fig. 1 Architecture of numerical computation environment

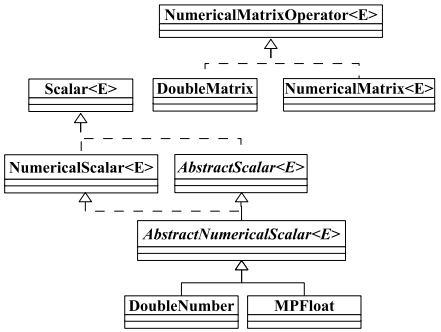


図 2 NFC の汎用数値データ型
Fig. 2 Architecture of numerical data type of NFC

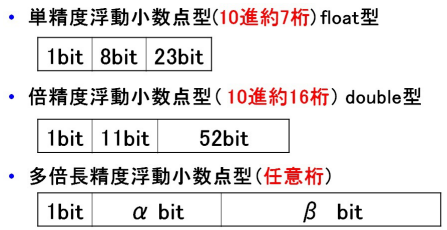


図 3 浮動小数点数のフォーマット
Fig. 3 Format of floating point number

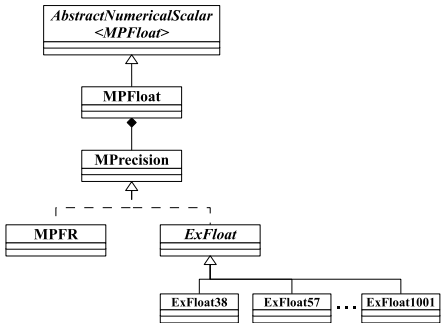


図 4 MPFloat のクラス図
Fig. 4 Class diagram of MPFloat

図 2 に NFC の数値データ型のクラス図を示す．NFC の行列演算や複素数演算は，汎用データ型である NumericalScalar に基づいて実装されているので，倍精度計算と多倍長計算を簡単に切り替えて，これらの演算を実行できる．また，ユーザプログラムを汎用的な数値データ型に対応させることで，プログラムをほとんど修正することなく倍精度計算，多倍長計算，精度保証付き数値計算，任意精度保証計算に対応させることができる．

3.1 多倍長計算

3.1.1 浮動小数点数のフォーマット

図 3 に浮動小数点数のフォーマットを示す．IEEE754 で定められた倍精度浮動小数点数は 1 ビットの符号部，11 ビットの指数部，52 ビットの仮数部を持っており，演算精度は 10 進数で約 16 桁である．多倍長精度浮動小数点数は，1 ビットの符号部をもち，指数部と仮数部を任意に指定することで，任意の演算精度を実現する．

3.1.2 多倍長演算パッケージ（MPFloat）

MPFloat では，GMP（MPFR）と Exflib の多倍長環境を切り替えて利用することができる．Exflib では，同時に複数の精度桁での計算を行うことができないので，桁数の異なる複数の Exflib のライブラリを準備し，MPFloat から適当な桁数のライブラリを利用し，桁数変換を行うことで，精度桁の動的な変更を実現した．MPFloat のクラス図を**図 4** に示す．ただし，ExFloat38 等の最後の数字は桁数を表す．

3.2 精度保証付き数値計算

3.2.1 精度保証付き数値計算の原理

実際の精度保証を実現する上で用いられている原理的手法は次の三つに要約される[14]．

(1) ベクトル内積の最大精度による計算
(2) 区間演算と不動点定理の適用による計算結果の数学的に完全な保証
(3) 反復残差改良法（iterative residual correction）による計算結果の高精度化

精度保証付き数値計算パッケージ CGA では，文献 2) で提案されている CPU の丸めモード変更による高速区間演算を採用している．また，区間の積に関しては文献 6) の区間の中心と半径から求める手法（Algorithm2.7 のスカラ版）を採用している．

3.2.2 精度保証付き数値計算パッケージ（CGA）

精度保証付き数値計算パッケージ CGA では，区間演算，区間行列演算[2),7),15)–17)]，関数の微分[2),15),18)]，線形方程式[2),7),15),17),18)]，非線形方程式[2),7),15),17)]，固有値問題[15),19)]，多項式の評価[18)] の精度保証付き数値計算を行うことができる．

CGA は NFC の汎用的な数値データ型に対応しているので，MPFloat と組み合わせて用いることが出来る．**図 5** に CGA のアーキテクチャ（パッケージ図）を示す．interval パッケージは区間演算，区間行列演算などを行うクラスを提供する．linear，nonlinear，eigen，polynomial，derivative の各パッケージは interval パッケージを利用して線形方程式，非線形方程式，固有値問題，多項式の評価，関数の微分の精度保証付き数値計算を行うクラスを提供する．そして，これらのパッケージの精度保証付き数値計算を行う主要クラスを**表 6** に示す．

表 6 CGA の主要クラス
Table 6 Main classes of CGA

精度保証付き数値計算	パッケージ	主要クラス
区間演算	interval	Interval
区間行列演算	interval	IntervalMatrix
関数の微分	derivative	IntervalDerivative
線形方程式	linear	LinearEquationVerifier
非線形方程式	nonlinear	NonLinearEquationVerifier
固有値問題	eigen	EigenVerifier
多項式の評価	polynomial	PolynomialVerifier

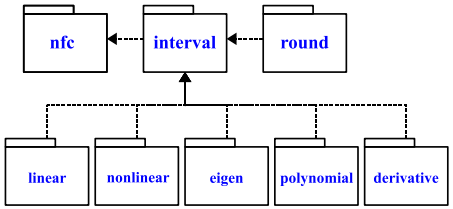


図 5 CGA のアーキテクチャ
Fig. 5 Architecture of CGA

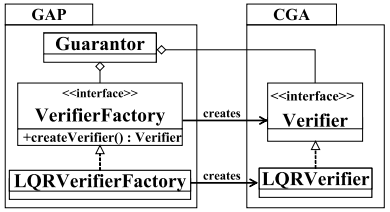


図 6 任意精度保証計算のためのクラス
Fig. 6 Architecture of GAP and CGA

3.3 任意精度保証計算パッケージ（GAP）

任意精度保証計算パッケージ GAP は，動的に演算精度を変更できる多倍長演算パッケージ MPFloat と，精度保証付き数値計算パッケージ CGA を組み合わせることで実現されている．**図 6** にこれらのパッケージの関係を示す．

CGA には精度保証付き数値計算を行うための Verifier インターフェースがあり，精度保証付き数値計算を行うクラスは，Verifier インターフェースを実装している．GAP には Verifier インスタンスを生成するファクトリーメソッドを持つ VerifierFactory インターフェースがある．指定された精度で計算結果を得る Guarantor クラスは，2 つのインターフェースを用いて定義されているので精度保証が可能な任意の計算に柔軟に対応できる．

4. 性 能 評 価

本章ではリカッチ方程式（P に関する行列方程式），

$$Q + A^T P + PA - PBR^{-1}B^T P = 0 \tag{12}$$

の解の導出に任意精度保証計算を適用して性能を評価する．ただし，文献 20) で提案されている次のパラメータを用いる．

$$V = I_3 - \frac{2}{3}vv^T,\ v^T = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$
$$A_0 = \varepsilon\,\mathrm{diag}\,(1,2,3),\ Q_0 = \mathrm{diag}\left(\frac{1}{\varepsilon},1,\varepsilon\right)$$
$$A = VA_0V,\ B = I_3, R = \varepsilon I_3,\ Q = VQ_0V$$

また，$\varepsilon = 10^{16}$ とする．

この問題を絶対誤差指定（1.0×10^{-8}）および精度桁指定（10 進 16 桁）によって評価する．比較のために，精度保証付き倍精度計算と精度保証付き多倍長計算（128bit）でも導出を行なう．実行環境を**表 7** に示す．以下では，演算時間は，演算を 10 回行った平均値である．

表 7 実行環境
Table 7 Computation environment

	環境
CPU	Intel Core2Duo E8400 3.00GHz
メモリ	4GB
OS	Windows XP Pro SP3
Java VM	Java(TM) SE Runtime Environment(build 1.6.0_15-b03)
GMP	Ver.4.3.1
MPFR	Ver.2.3.2
MPFR C++[21]	Ver.2.1

4.1 精度保証付き倍精度計算

方程式の係数行列を倍精度で作成するプログラムを以下に示す．

方程式の係数行列（倍精度）

```
double e = 1.0e18;
double[] a0 = new double[] {1,2,3};
double[] q0 = new double[] {1/e,1,e};
Matrix A0 = DiagonalMatrix.create(a0).multiply(e);
Matrix Q0 = DiagonalMatrix.create(q0);
Matrix I = A0.createUnit(3);
Matrix V = I.subtract(A0.createOnes(3).multiply(2).divide(3));
Matrix A = V.multiply(A0).multiply(V);
Matrix B = I;
Matrix Q = V.multiply(Q0).multiply(V);
Matrix R = I.multiply(e);
```

また，精度保証付き数値計算を行うためのプログラムを以下に示す．ただし，LQRVerifier2 は CGA が提供するリカッチ方程式の解の精度保証付き数値計算[13),15),19)] を行うクラスである．

精度保証付き数値計算

```
Verifier verifier = new LQRVerifier2(A,B,Q,R);
verifier.setUsingKrawczyk(false);
verifier.solve();
```

この問題では，倍精度では誤差の影響により数値計算が破綻し，解を求めることができなかった．

4.2 精度保証付き多倍長計算

方程式の係数行列を多倍長精度で作成するプログラムを以下に示す．

CGA の各クラスは，NFC の汎用的な数値データ型に対応しているので，多倍長精度で作成したこれらの行列をそのまま，4.1 節で示した精度保証付き数値計算を行うプログラムに渡すことができる．

方程式の係数行列（多倍長精度）

```
MPFloat e = new MPFloat("1.0e18");
MPFloat[] a0 = new MPFloat[] {new MPFloat(1), new MPFloat(2), new MPFloat(3)};
MPFloat[] q0 = new MPFloat[] {new MPFloat(1).divide(e), new MPFloat(1), e};
Matrix A0 = DiagonalMatrix.create(a0).multiply(e);
Matrix Q0 = DiagonalMatrix.create(q0);
Matrix I = A0.createUnit(3);
Matrix V = I.subtract(A0.createOnes(3).multiply(2).divide(3));
Matrix A = V.multiply(A0).multiply(V);
Matrix B = I;
Matrix Q = V.multiply(Q0).multiply(V);
Matrix R = I.multiply(e);
```

精度保証付き多倍長計算（仮数部 128bit）で方程式の解を求めると，以下の値が得られた．

```
===  P 中心  (   3  x   3)  NumericalMatrix  ===
      [   (  1)  ]    [   (  2)  ]    [   (  3)  ]
(  1)   4.667e+32       1.333e+32      -3.704e-2
(  2)   1.333e+32       4e+32          -1.333e+32
(  3)  -3.704e-2       -1.333e+32       3.333e+32
```

```
===  P 半径  (   3  x   3)  NumericalMatrix  ===
      [  (  1)  ]    [  (  2)  ]    [  (  3)  ]
(  1)   1.278e-4       1.14e-4        1.217e-4
(  2)   1.354e-4       1.24e-4        1.202e-4
(  3)   1.103e-4       1.073e-4       1.03e-4
```

このとき，各成分の中心の有効桁数は，

```
===  有効桁数  (   3  x   3)  IntMatrix  ===
             (  1)       (  2)       (  3)
(  1)          37          37           3
(  2)          36          37          37
(  3)           3          37          37
```

である．有効桁数が 3 桁しかない成分があることが分かる．

このように，精度保証付き多倍長計算では誤差の大きさを知ることができるが，適切に仮数部のビット長を指定しないと，有効桁数がほとんどない結果となる場合がある．そのような場合には，ビット長を選びなおし，計算をやり直す必要がある．

4.3 絶対誤差指定

以下に，2.4 節に示した手法を用いて，絶対誤差を指定して任意精度保証計算を行なうプログラムを示す．行列 A, B, Q, R は，4.2 節で作成された行列と同じである．

任意精度保証計算（絶対誤差）のプログラム

```
VerifierFactory factory = new LQRVerifier2Factory(A, B, Q, R);
Guarantor guarantor = new Guarantor(factory);
guarantor.solveWithErrorBound(1e-8);
```

絶対誤差を 1.0×10^{-8} と指定すると，以下のような解が求まった．

```
===  P 中心  (   3  x   3)  NumericalMatrix  ===
      [   (  1)  ]    [   (  2)  ]    [   (  3)  ]
(  1)   4.667e+32       1.333e+32      -3.704e-2
(  2)   1.333e+32       4e+32          -1.333e+32
(  3)  -3.704e-2       -1.333e+32       3.333e+32

===  P 半径  (   3  x   3)  NumericalMatrix  ===
      [   (  1)  ]    [   (  2)  ]    [   (  3)  ]
(  1)   8.731e-10       8.622e-10       8.363e-10
(  2)   1.04e-9         1.019e-9        9.459e-10
(  3)   8.93e-10        8.367e-10       8.149e-10
```

全ての成分の半径が，指定した絶対誤差（1.0×10^{-8}）よりも小さいことを確認できる．このときの各成分の中心の有効桁数を求めると，

```
===  有効桁数  (   3  x   3)  IntMatrix  ===
             (  1)       (  2)       (  3)
(  1)          42          42           8
(  2)          42          42          42
(  3)           8          42          42
```

となった．絶対誤差指定の場合，成分によっては，有効桁数が小さくなることがあるので，注意が必要であるが，真の解の大きさがある程度予測できる場合には有効な方法である．

4.4 精度桁指定

以下に，2.3 節に示した手法を用いて，精度桁を指定して任意精度保証計算を行なうためのプログラムを示す．行列 A, B, Q, R は，4.2 節で作成された行列と同じである．

任意精度保証計算（精度桁）のプログラム

```
VerifierFactory factory = new LQRVerifier2Factory(A, B, Q, R);
Guarantor guarantor = new Guarantor(factory);
guarantor.solvewithEffectiveDigits(16);
```

精度桁を 16 桁（10 進数）と指定すると，以下のような解が求まった．ただし，10 進数で指定された精度桁は内部で 2 進数に変換される．

```
===  P 中心  (   3  x   3)  NumericalMatrix  ===
      [   (  1)  ]    [   (  2)  ]    [   (  3)  ]
(  1)   4.667e+32       1.333e+32      -3.704e-2
(  2)   1.333e+32       4e+32          -1.333e+32
(  3)  -3.704e-2       -1.333e+32       3.333e+32

===  P 半径  (   3  x   3)  NumericalMatrix  ===
      [   (  1)  ]    [   (  2)  ]    [   (  3)  ]
(  1)   2.141e-18       2.019e-18       1.949e-18
(  2)   1.701e-18       1.626e-18       1.687e-18
(  3)   1.439e-18       1.403e-18       1.355e-18
```

このとき区間の中心の有効桁数（10 進数）を求めると，

```
===  有効桁数  (   3  x   3)  IntMatrix  ===
             (  1)       (  2)       (  3)
(  1)          51          50          17
(  2)          50          51          50
(  3)          17          50          51
```

となった．これより，得られた解の有効桁数は指定した 16 桁よりも大きいことを確認できる．

表 8 演算方法と計算に用いられた仮数部のビット長
Table 8 Bit-length of Mantissa for calculation

演算方法	1 回目	2 回目
精度保証付き倍精度演算⋆1	53bit	-
精度保証付き多倍長演算	128bit	-
任意精度保証計算（絶対誤差）	128bit	145bit
任意精度保証計算（精度桁）	128bit	174bit

表 9 演算時間
Table 9 Time for Execution

演算方法	演算時間 [s]
精度保証付き倍精度演算⋆2	-
精度保証付き多倍長演算	0.625
任意精度保証計算（絶対誤差）	1.293
任意精度保証計算（精度桁）	1.298

4.5 仮数部のビット長

上述の演算時で用いられた仮数部のビット長を**表 8** に示す．これより，両方の任意精度保証計算において，2 回目の演算で指定された精度を満たすことができ，演算を終えていることが分かる．

また，これらの演算に要した時間を**表 9** に示す．任意精度保証計算では，精度保証付き多倍長計算を 2 回行うので，精度保証付き多倍長計算の 2 倍程度の演算時間になっている．

5. おわりに

本論文では，多倍長演算における仮数部のビット長を適応的に調節することで任意の結果精度を保証できる手法を提案した．そして，提案手法に基づき数値計算パッケージを開発し，具体的な問題に適用し評価した．

大石らは，Knuth や Veltkamp-Dekker の定理を用いて，計算精度を段階的に上げながら必要な精度の解を倍精度浮動小数点数で計算する手法を提案している[1),22)]．この手法を用いると，新たに多倍長演算ライブラリを導入しなくても高精度な計算を行うことができる．しかし，この方法は段階的に精度を上げていく必要があるため，本研究で提案した 2 回の多倍長演算で解が得られる手法よりも計算量が多くなる場合があると考えられる．

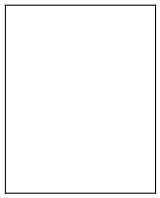
謝辞 本研究は，科学研究費補助金（基盤 C：21560467）の助成を受けたものである．ここに謝意を示します．

参 考 文 献

1) 荻田武史：品質を落とさない数値計算法–無誤差変換と高精度計算 (特集計算の品質–精度・誤差・効率)，数学セミナー， Vol.47, No.11, pp.15–19 (2008/11).

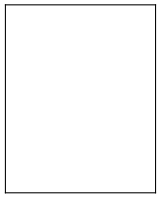
⋆1 実際には数値計算が破綻してしまう．
⋆2 同上

2) 大石進一：精度保証付き数値計算，コロナ社 (2000).
3) 坂口秀雄，渡部義隆，今井仁司：多倍長計算を適用した精度保証数値計算，数理解析研究所講究録， Vol.1441, pp.165–172 (2005).
4) 山本野人， 松田望：多倍長演算を利用した Bessel 関数の精度保証付き数値計算，日本応用数理学会論文誌， Vol.15, No.3, pp.347–359 (2005).
5) 小郷 寛，美多 勉：システム制御理論入門，実教出版 (1979).
6) Rump, S. M.: INTLAB-INTERVAL LABORATORY, *Developments in Reliable Computing (edited by Csendes, T.)*, pp.77–104 (1999).
7) 中尾充宏，山本野人：精度保証付き数値計算，日本評論社 (1998).
8) 藤原宏志：Multiple-Precision Arithmetic Library exflib (2006). http://www-an.acs.i.kyoto-u.ac.jp/~fujiwara/exflib/.
9) : GMP web pages. http://gmplib.org/.
10) 古賀雅伸， 松木毅：OS 中立な数値計算ライブラリの開発と制御系設計への適用，計測自動制御学会制御部門大会， Vol.3, pp.725–728 (2003).
11) 山村英介，古賀雅伸，矢野健太郎，山田賢治：多倍長計算を用いた制御系設計パッケージ，第 52 回システム制御情報学会研究発表講演会 (2008).
12) : The MPFR Library. http://www.mpfr.org/.
13) 矢野健太郎，古賀雅伸：LQ 制御問題の精度保証付き数値計算，計測自動制御学会論文集， Vol.45, No.5, pp.261–267 (2009).
14) 中尾充宏：精度保証付き数値計算の現状と動向，情報処理， Vol.31, No.9, pp.1177–1190 (1990).
15) 日本数学会：218 精度保証付き数値計算法. 岩波数学辞典第 4 版，岩波書店, pp. 642-647 (2007).
16) Moore, R.E.: *Interval Analysis*, Prentice-Hall, Englewood Cliffs, NJ (1966).
17) Hargreaves, G.I.: *Interval Analysis in MATLAB* (2002). Numerical Analysis Report No.416.
18) 大石進一：数値計算，裳華房 (1999).
19) Rump, S.M.: Computational error bounds for multiple or nearly multiple eigenvalues, *Linear Algebra and its Applications*, Vol.324, pp.209–226 (2001).
20) Benner, P., Laub, A. and Mehrmann, V.: A Collection of Benchmark Examples for the Numerical Solution of Algebraic Riccati Equations I: Continious-Time Case (1995). Tech. Report SPC 95 22, Fak. f. Mathematik, TU Chemnitz-Zwickau, 09107 Chemnitz, FRG.
21) : MPFR C++. http://www.holoborodko.com/pavel/?page_id=12.
22) 大石進一：なぜ精度保証付き数値計算の研究を追求したか，Fundamentals Review，電子情報通信学会基礎境界ソサイエティ誌 Vol. 2，No.2 (2008) pp.9-19. コロナ社 (2008).



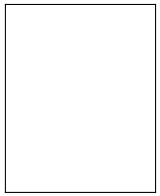
中島 大雅

2009 年九州工業大学情報工学部システム創成情報工学科卒業．同年同大学院情報工学府情報創成工学専攻修士課程入学後，現在に至る．多倍長精度演算に関する研究に従事．情報処理学会の学生会員．



古賀 雅伸

1993 年東京工業大学大学院理工学研究科制御工学専攻博士課程修了．博士（工学）．東京工業大学工学部助手，同大学情報理工学研究科助手，九州工業大学情報工学部助教授を経て，現在同大学大学院情報工学研究院准教授．制御系 CAD，科学技術計算基盤に関する研究に従事．IEEE，計測自動制御学会等の会員．



矢野健太郎

2009 年九州工業大学大学院情報工学研究科情報創成工学専攻博士後期課程単位取得満期退学．同年同大学大学院情報工学研究院産学官連携研究員．2010 年福岡工業大学短期大学部情報メディア学科特任教員助教となり，現在に至る．精度保証付き数値計算に基づく検証付き制御系設計，リアルタイム制御に関する研究に従事．博士（情報工学）．計測自動制御学会，電気学会などの会員．