制御系設計における多倍長計算

Multiple-Precision Arithmetic for Control System Design

九州工業大学 ○中島 大雅，古賀 雅伸, 矢野 健太郎

○ Hiromasa Nakashima and Masanobu Koga and Yano Kentarou

Kyusyu Institute of technology

Abstract This study develops the Multiple-Precision Arithmetic package and the numerical computation package which can get a calculation result with the precision that was appointed by regulating the number of the digits for multiple-precision arithmetic automatically. We verify it by applying these packages to the system control design such as SDPJ and LQ control problem.

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，丸めなどの誤差が混入するため，計算機上では正確な結果を得られるとは限らない。また、精度桁を任意に設定できる多倍長精度計算を用いれば丸め誤差を低減することができるが、誤差の大きさがどの位あるのかはわからない。一方，精度保証計算 [1] を用いれば、誤差の大きさを見積もることができる。しかし精度保証計算は、事後的に誤差を保証するものであり、事前に保証することはできない。計算方法と誤差の関係を表 1 に示す.

表 1: 計算方法と誤差

計算方法	誤差の大きさ	誤差保証
単・倍精度	大	不可
精度保証付き倍精度	大	可 (事後保証)
多倍長	小	不可
精度保証付き多倍長	小	可 (事後保証)
???	小	可 (事前保証)

本研究では、多倍長計算を行うパッケージ、および、多倍長計算の精度桁を自動調節することで指定された精度で計算結果を得ることができる数値計算パッケージを開発する．そして，多倍長計算と精度指定可能な数値計算を、SDP(半正定値計画問題)、極配置問題、LQ 最適制御問題などの制御系設計に適用し，その有効性を検証する．

2 数値計算環境

近年，GMP(C)[2]，ExFlib(C++)[3]，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，その有効性が期待されている。しかしながら，多倍長計算を各分野の計算に適用するには既存のソフトウェアを修正・改良する時間と手間がかかるため，その適用事例の報告は多くない.

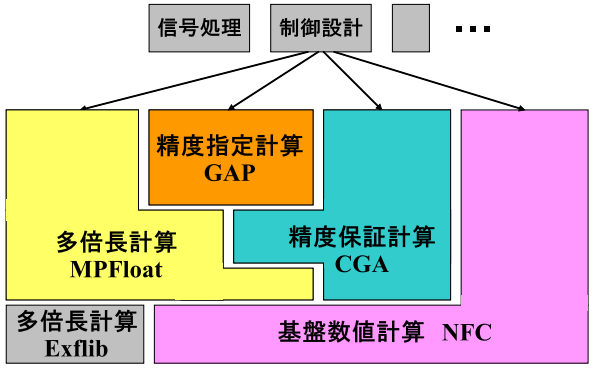

図 1: 数値計算環境の構造

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[4] 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 (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[5] は，C++で実装された多倍長演算パッケージ ExFlib[3] を Java から利用するためのパッケージである．NFC の汎用データ型に対応することで多倍長精度での数値計算を実現している．精度保証計算パッケージ CGA(Computing with Guranteed Accuracy)[6] は，倍精度での精度保証付き計算及び MPFloat を用いた多倍長精度での精度保証付き計算を行うことができる．今回開発した精度指定計算パッケージ GAP(Guranteed a Priori Precision) では、CGA と MPFloat を用いることで精度指定可能な演算を実現している．

制御設計を行うソフトウェアを NFC の汎用的な数値データ型に対応させることで，ソフトウェアをほとんど修正することなく倍精度計算、多倍長計算、精度保証計算、精度指定計算を行うことができる。

2.1 多倍長計算

2.1.1 多倍長計算の原理

多倍長計算とは，浮動小数点数の仮数部の大きさを変えることで丸め誤差の影響を抑えようとする数値計算である．図 2 に浮動小数点数のフォーマットを示す．倍精度浮動小数点数は 1 ビットの符合部，11 ビットの指数部，52 ビットの仮数部を持っており，10 進数で約 16 桁の精度を持つ．多倍長精度浮動小数点数は，1 ビットの符合部を持ち，指数部と仮数部を任意に指定できるので，任意の精度を実現できる．

- 単精度浮動小数点型(10進約7桁) float型

1bit	8bit	23bit

- 倍精度浮動小数点(10進約16桁) double型

1bit	11bit	52bit

- 多倍長精度浮動小数点型(任意桁型)

1bit	αbit	βbit

図 2: 浮動小数点数のフォーマット

2.1.2 多倍長演算パッケージ

本研究では，多倍長演算 MPFloat について動的に精度桁を変更できるように改良した．MPFloat が利用する ExFlib では，同時に複数の精度桁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，精度桁がの動的な変更を実現した．

2.2 精度保証付き数値計算 [6]

2.2.1 精度保証付き数値計算の原理

ここでは問題の解を精度保証付きで求めるための原理と手法をごく一般的な形において説明する [7].

精度保証付き数値計算は以下に示す手順で行われる．

1. 真の解を含む集合の候補（候補集合）を何らかの方法で決定
2. 候補集合が真の解を含むことを（不動点定理 [7] を用いて）検証
3. 真の解を含まなければ，別の候補（集合）を立てて，前段に戻る
4. 真の解を含んでいれば，解の精度がその集合の大きさによって決定され，計算終了

2.2.2 精度保証付き数値計算パッケージ

我々が開発した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，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 3 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．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．そして，これらのパッケージは主要クラスとして，表 2 に示す精度保証計算を行うクラスを含んでいる．

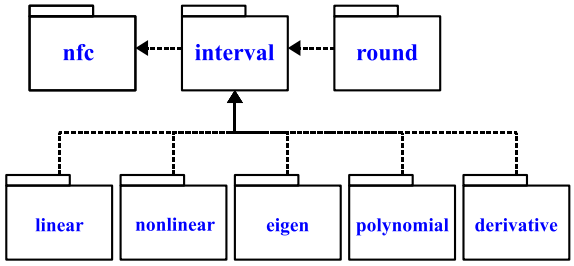

図 3: CGA のアーキテクチャ

表 2: CGA の主要クラス

精度保証計算	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

3 精度指定可能な数値計算

3.1 精度桁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について述べる．次のシステムの極配置問題について考える．

$$\dot{x} = Ax + Bu$$
$$u = -Fx$$

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

$$\text{指定極} = \{-1, -5, -10, -15, -20, -25\}$$

閉ループ系 $\dot{x} = (A - BF)x$ の極が指定極となるような状態フィードバックゲイン行列 F を求める。

上に示す極配置問題 ($e = 10^5$, $e = 10^{10}$) を複数の精度桁で精度保証付き多倍長演算を用いて解き，それぞれの失われた精度の桁数を求めた．その結果を表 3 と表 4 に示す．ただし，最小の有効桁数とは，求めた区間 F の各成分の有効桁数の最小値である．有効桁数 [8] は式 (1) によって求める．

$$\text{有効桁数} = \log_{10} \left| \frac{1}{\text{相対誤差}} \right| \tag{1}$$

表 3: 精度桁と失われた精度の桁数 ($e = 10^5$)

精度桁数	最小の有効桁数	最大の失われた精度の桁数
115 桁	110 桁	**5 桁**
211 桁	206 桁	**5 桁**
308 桁	302 桁	**6 桁**
404 桁	399 桁	**5 桁**
500 桁	495 桁	**5 桁**

表 4: 精度桁と失われた精度の桁数 ($e = 10^{10}$)

精度桁数	最小の有効桁数	最大の失われた精度の桁数
115 桁	105 桁	**10 桁**
211 桁	201 桁	**10 桁**
308 桁	397 桁	**11 桁**
404 桁	394 桁	**10 桁**
500 桁	490 桁	**10 桁**

この結果より，仮数部のビット長と失われる精度の桁数にはほとんど関係がないことが分かった。他の計算についても調べたところ，同様の結果を得ることができた．このことから，仮数部のビット長と失われる精度の桁数は関係ないと考えられる．

3.2 指定精度の実現

GAP は CGA が求めた解が保証する精度の桁数が，指定された精度を満たすか判定し，満たしていない場合、適当に桁数を増やして、再度計算を行うという動作を繰り返す。

精度を指定する方法として、絶対誤差指定 [8] と精度桁指定 [8] が選択可能である。以下の式に基づいて、精度保証付き計算に用いる桁数 (p_k) を更新する。

- **精度桁指定 (p^*:(2 進数))**

$$p_{k+1} = p_k + \Delta p_k \tag{2}$$
$$p_0 = \lceil p^*/N \rceil \times N \tag{3}$$
$$\Delta p_k = \lceil (p^* - \hat{p}_k) \times a/N \rceil \times N \tag{4}$$
$$\hat{p}_k = \log_2 |c_k/r_k| \tag{5}$$

- **絶対誤差指定 (e^*:(10 進数))**

$$p_{k+1} = p_k + \Delta p_k \tag{6}$$
$$p_0 = N \tag{7}$$
$$\Delta p_k = \lceil (p^* - \hat{p}_k) \times a/N \rceil \times N \tag{8}$$
$$\hat{p}_k = \log_2 |c_k/r_k| \tag{9}$$
$$p_k^* = \log_2 |c_k/e^*| \tag{10}$$

ただし，p^* は指定された精度の 2 進数時の桁数，p_k は使用桁数 (2 進数)、$\hat{p}_k$ は保証桁数 (2 進数),N は増加するビット長の単位 (64)、c_r は中心,r_k は半径,a はゲインである。また、$\lceil \cdots \rceil$ は上向きに丸めた整数を求める関数である。

(3) 式と (7) 式は初期桁数を決定する．(4) 式と (5) 式では，精度保証で求まった値の中心と半径からその値の有効桁数を求め，指定された桁数との差をとることで増加させる桁数を求める．(8) 式，(9) 式，(10) 式は精度保証で求まった値の半径と指定された絶対誤差から増加させる桁数を求める．(2)，(6) 式は指定された精度を満たすために足りない桁数を求め，現在使用している桁数に加えることで次に使用する桁数を求める．

3.1 節の結果より，仮数部のビット長と失われる精度の桁数は関係ない考えられるので，多くの場合に 2 回目の演算には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る．

3.3 精度指定可能な数値計算パッケージ

2 章で述べたように，精度指定可能な数値計算をサポートするパッケージ GAP は，動的に精度桁を変更できる多倍長計算パッケージ 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 4 にこれらの関係を示す。

CGA には精度保証付き計算を行うための Verifie インターフェースがある。精度保証付き計算を行うクラス

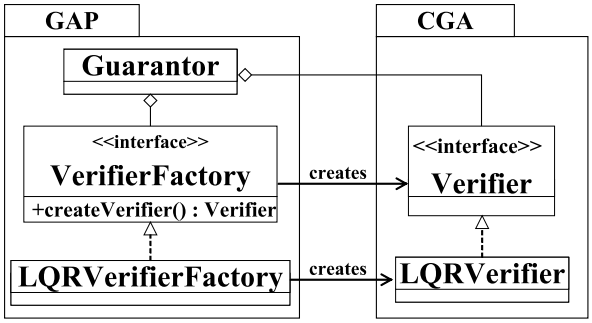

図 4: 精度指定計算のためのクラス図

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に基づ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対応できる。

4 性能評価

本章では多倍長演算，精度保証付き計算，精度指定付き計算を SDP(Semidefinite Program)，リカッチ方程式の解の導出などに適用して，これらの有効性を示す．

実行環境を表 5 に示す。なお，演算時間は，演算を 10 回行った平均値である．

表 5: 実行環境

	環境
CPU	Intel Core2Duo E8400
メモリ	4GB
OS	Windows XP Pro SP3
Java VM	Java(TM) SE Runtime Environment (build 1.6.0_15-b03)

4.1 多倍長演算を用いた SDP

SDPJ[9] は SDPA[10] に基づき我々が開発した Java 言語による半正定値計画問題に対するソルバーである．数値のデータ型に依存することなく実行することができるため，倍精度計算，多倍長計算に対応している．

双対ギャップ

半正定値計画問題は双対定理より，主問題および双対問題のそれぞれに内点許容解が存在すれば，最適値は一致し，最適解が存在する．このとき，主問題の最適値と双対問題の最適値との間に差が生じる場合、その差を双対ギャップという．

SDPJ で SDPLIB[11] のデータファイル「hinf2.dat-s」を用いて検証する．この問題の制約行列の数は 13，ブロック数は 3，1 つの行列の次数は 16 である．

倍精度・多倍長精度でそれぞれ実行した結果の主問題と相対問題の解の相対ギャップ，実行時間を表 6 に示す．

表 6: 相対ギャップと実行時間

演算方法	相対ギャップ	演算時間 [s]
SDPJ(倍精度)	1.171e-05	0.22
SDPJ(115 桁)	1.851e-116	2.27

倍精度では，解は実行可能領域にはあるものの、18 回目の反復で内点法の反復が異常終了した。しかし、多倍長精度では終了条件を満たして反復を正常終了し，仕様を満たす最適解を求めることができた。

4.2 精度保証付き多倍長演算

文献 [12] で提案されている次のリカッチ方程式の解を求める．

$$Q + A^T P + PA - PGP = 0 \tag{11}$$

ただし，

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

である．

上のリカッチ方程式 (11) の解を精度指定で求め性能評価を行う。$\varepsilon = 10^6$ とし、精度保証付き倍精度演算と，精度保証付き多倍長演算 (38 桁) によって精度評価する。

4.2.1 精度保証付き倍精度計算

以下に精度保証付き倍精度演算を用いて解く場合のプログラムを示す．倍精度の行列である DoubleMatrix 型で A, B, Q, R を定義し，LQRVerifier2 に引数で渡すことで，リカッチ方程式の解を精度保証付き倍精度演算で求めることができる．

精度保証付き倍精度演算を行うプログラム

```
Matrix A0, Q0, V, A, B, Q, R;
double[] a0 = new double[] {1, 2, 3};
double[] q0 = new double[] {1 / e, 1, e};
A0 = DoubleMatrix.diagonal(a0).multiply(e);
Q0 = DoubleMatrix.diagonal(q0);
V = A0.createUnit(3)
   .subtract(A0.createOnes(3).multiply(2).divide(3));
A = V.multiply(A0).multiply(V);
B = V.createUnit(3);
Q = V.multiply(Q0).multiply(V);
R = V.createUnit(3).multiply(e);
LQRVerifier2 verifier = new LQRVerifier2(A, B, Q, R);
verifier.setUsingKrawczyk(false);
verifier.solve();
```

精度保証付き倍精度演算を用いて求めたリカッチ方程式の解は，

```
===  P 中心  (   3  x   3)  Matrix  ===
         (  1)          (  2)          (  3)
(  1)  4.667e+12      1.333e+12     -3.711e-02
(  2)  1.333e+12      4.000e+12     -1.333e+12
(  3) -3.760e-02     -1.333e+12      3.333e+12
===  P 半径  (   3  x   3)  Matrix  ===
         (  1)          (  2)          (  3)
(  1) 3.906e-03     4.150e-03     4.287e-03
(  2) 3.906e-03     3.906e-03     4.150e-03
(  3) 3.678e-03     3.418e-03     3.906e-03
```

となる．成分毎に，中心行列の値を中心とし半径行列の値を半径とする区間内にリカッチ方程式の解が存在することが保証される．各成分の中心の有効桁数を求めると，

```
===  有効桁数  (   3  x   3)  IntegerMatrix  ===
             (  1)        (  2)        (  3)
(  1)          15           14            0
(  2)          14           15           14
(  3)           1           14           14
```

となる．この結果をみると，有効桁がほとんどない成分があることが確認できる．このことから，この値がほとんど信用できないことがわかる．また，この解の中心を (11) 式に代入すると，

```
===  残差中心  (   3  x   3)  Matrix  ===
         (  1)          (  2)          (  3)
(  1) -2.048e+03     -2.048e+03     -1.792e+03
(  2) -2.048e+03      2.048e+03      1.024e+03
(  3)  5.120e+02      2.048e+03     -4.096e+03
===  残差半径  (   3  x   3)  Matrix  ===
         (  1)          (  2)          (  3)
(  1) 7.168e+04     7.578e+04     6.810e+04
(  2) 7.373e+04     7.578e+04     6.861e+04
(  3) 5.862e+04     6.144e+04     5.325e+04
```

という結果となり，区間に 0 を含んではいるが，かなり大きな値となっている．

4.2.2 精度保証付き多倍長計算

以下に精度保証付き多倍長演算を用いて解く場合のプログラムを示す．LQRVerifier2 は汎用的な数値データ型に対応しているので，行列 A, B, Q, R を多倍長精度の行列である NumericalMatrix <MPFloat> 型で定義することで，多倍長精度で求めることができる．

精度保証付き多倍長演算を行うプログラム

```
Matrix A0, Q0, V, A, B, Q, R;
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};
A0 = BaseMatrix.diagonal(a0).multiply(e);
Q0 = BaseMatrix.diagonal(q0);
V = A0.createUnit(3)
   .subtract(A0.createOnes(3).multiply(2).divide(3));
A = V.multiply(A0).multiply(V);
B = V.createUnit(3);
Q = V.multiply(Q0).multiply(V);
R = V.createUnit(3).multiply(e);
LQRVerifier2 verifier = new LQRVerifier2(A, B, Q, R);
verifier.setUsingKrawczyk(false);
verifier.solve();
```

精度保証付き多倍長演算を用いてリカッチ方程式の解は，

```
===  P 中心  (   3  x   3)  NumericalMatrix  ===
       [   (  1)  ]    [   (  2)  ]    [   (  3)  ]
(  1)   4.667e+12       1.333e+12      -3.704e-2
(  2)   1.333e+12       4.000e+12      -1.333e+12
(  3)  -3.704e-2       -1.333e+12       3.333e+12
===  P 半径  (   3  x   3)  NumericalMatrix  ===
       [   (  1)  ]    [   (  2)  ]    [   (  3)  ]
(  1)   8.013e-25       7.593e-25       6.858e-25
(  2)   6.689e-25       6.592e-25       6.365e-25
(  3)   6.253e-25       6.010e-25       6.010e-25
```

となる．この各成分中心の有効桁数を求めると，

```
===  有効桁数  (   3  x   3)  IntegerMatrix  ===
             (  1)        (  2)        (  3)
(  1)          36           36           22
(  2)          36           36           36
(  3)          22           36           36
```

となる．この結果をみると，最小でも 22 桁であるので，十分な有効桁数を保てていることがわかる．また，この解の中心を (11) 式に代入すると，

```
===  残差中心  (   3  x   3)  NumericalMatrix  ===
       [   (  1)  ]    [   (  2)  ]    [   (  3)  ]
(  1)  -1.505e-19       -1.441e-19       1.151e-20
(  2)  -1.484e-19       -2.972e-20       4.381e-20
(  3)   4.027e-21       -1.352e-19       8.196e-21
===  残差半径  (   3  x   3)  NumericalMatrix  ===
       [   (  1)  ]    [   (  2)  ]    [   (  3)  ]
(  1)   1.400e-17       1.402e-17       1.103e-17
(  2)   1.260e-17       1.310e-17       1.102e-17
(  3)   1.000e-17       1.035e-17       8.500e-18
```

という結果となり，区間に 0 を含み，かなり小さい値となる．

このように，多倍長演算と精度保証付き計算を組み合わせることにより，高い精度が保証された解を求めることができる．

これらのリカッチ方程式の解の導出にかかった演算時間を表 7 に示す．

表 7: 演算時間

演算方法	演算時間 [s]
精度保証付き倍精度演算	0.033
精度保証付き多倍長演算	0.328

4.3 精度指定付き多倍長演算

4.2 節で述べた問題を，精度指定で求め性能評価を行う。$\varepsilon = 10^{19}$ とし、絶対誤差指定 (1.0×10^{-16}) 及び精度桁指定 (20 桁) によって精度評価する。

絶対誤差指定

絶対誤差を 1.0×10^{-16} として，リカッチ方程式の解を求めると，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+38       1.333e+38       -3.704e-2
(  2)   1.333e+38       4.000e+38       -1.333e+38
(  3)  -3.704e-2       -1.333e+38        3.333e+38
===  P 半径  (   3  x   3)  NumericalMatrix  ===
      [   (  1)  ]    [   (  2)  ]    [   (  3)  ]
(  1)   3.253e-18       3.009e-18       2.877e-18
(  2)   3.442e-18       3.415e-18       3.104e-18
(  3)   2.963e-18       2.778e-18       2.765e-18
```

全ての成分の半径が、指定した絶対誤差 (1.0×10^{-16}) 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)  IntegerMatrix  ===
             (  1)       (  2)       (  3)
(  1)          56          55          16
(  2)          55          56          55
(  3)          16          55          56
```

となる．

絶対誤差指定の場合，中心の値によっては，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．真の解の大きさがある程度予測できる場合は有効である．

精度桁指定

精度桁を 20 桁としてリカッチ方程式の解を求めると，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+38       1.333e+38       -3.704e-2
(  2)   1.333e+38       4.000e+38       -1.333e+38
(  3)  -3.704e-2       -1.333e+38        3.333e+38
===  P 半径  (   3  x   3)  NumericalMatrix  ===
      [   (  1)  ]    [   (  2)  ]    [   (  3)  ]
(  1)   1.616e-37       1.477e-37       1.403e-37
(  2)   1.734e-37       1.675e-37       1.484e-37
(  3)   1.543e-37       1.440e-37       1.440e-37
```

このときの区間の中心の有効桁数を求めると，

```
===  有効桁数  (   3  x   3)  IntegerMatrix  ===
             (  1)       (  2)       (  3)
(  1)          75          74          35
(  2)          74          75          74
(  3)          35          74          75
```

となる．これより，求めた解の有効桁数は指定した 20 桁よりも大きいことを確認できる．

これらのリカッチ方程式の解の導出にかかる演算時間を表 8 に示す．

表 8: 演算時間

演算方法	演算時間 [s]
精度指定付き多倍長演算 (絶対誤差指定)	0.765
精度保証付き多倍長演算 (精度桁指定)	0.731

5 まとめ

本研究では、多倍長計算を行うパッケージ、および、多倍長計算の精度桁を自動調節することで指定された精度で計算結果を得ることができる数値計算パッケージを開発した．そして，多倍長計算と、精度指定可能な数値計算を、SDP(半正定値計画問題)、極配置問題、LQ 最適制御問題などの制御系設計に適用し，その有効性を検証した．

今後は，並列処理機能を用いて速度向上を図りたい．

参考文献

[1] 大石進一. 精度保証付き数値計算. コロナ社, 2000.

[2] Gnu mp. http://gmplib.org/.

[3] 藤原宏志. Multiple-Precision Arithmetic Library exflib(C++), 2006.

[4] 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. 第 3 回計測自動制御学会制御部門大会, 2003.

[5] 山村英介. 多倍長精度計算を用いた制御系設計パッケージ. 卒業論文, 2007.

[6] 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 第 45 巻 第 5 号 pp261-267, 2009.

[7] 中尾充宏, 山本野人. 精度保証付き数値計算 チュートリアル：応用数理最前線. 日本評論社, 1998.

[8] 高倉葉子. 数値計算の基礎 - 解放と誤差 - . コロナ社, 2007.

[9] 古賀崇史, 古賀雅伸. 多倍長計算を用いた半正定値計画問題に基づく制御系設計支援パッケージの開発. 平成 20 年度 修士論文, 2009.

[10] SDPA(SemiDefiniteProgrammingAlgorithm). http://grid.r.dendai.ac.jp/sdpa/.

[11] BRIAN BORCHERS. SDPLIB 1.2, A Library of Semidefinite Programming Test Problems, 1999. http://infohost.nmt.edu/ sdplib/.

[12] P. Benner, A. Laub, and V. Mehrmann. 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.