Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

LQ Control Problem based on Numerical Computation with Guaranteed Accuracy

Kentaro Yano [*] **Masanobu Koga** [*]

[*] *Kyushu Institute of Technology, 680-4 Kawazu Iizuka Fukuoka, Japan (Tel: +81-948-29-7726; e-mail: yano@mk.ces.kyutech.ac.jp, koga@ces.kyutech.ac.jp).*

Abstract: This paper presents LQ control problem based on numerical computation with guaranteed accuracy. By using the proposed methods, it is possible to guarantee numerical quality about design of control system. This paper also proposes a problem which finds the numerical optimal controller from a set of solutions solved by verified LQ control problem.

Keywords: Design of control system, numerical computation with guaranteed accuracy, LQ control problem, Riccati equation, Genetic Algorithms

1. INTRODUCTION

A computer is used for various areas now, but it was devised that a real number is approximated to a floating point number and computed to compute fast. So, a usual numerical computation using a computer cannot solve a mathematically exact solution because of a round error etc. Many mathematicians and researchers took notice of this error, but it was rarely estimated how much an error is big correctly until recently.

To solve this issue, Numerical Computation with Guaranteed Accuracy(Oishi (2000a)) which computes an error occurred by a numerical computation using a computer, and proves how much a computational result is correct was appeared. In various areas, there are great hopes that apply this computational technique to those areas.

Recently, methods are proposed which apply a numerical computation with guaranteed accuracy to design of a control system (Kanno and Hara (2007); Yano and Koga (2007)). For example, a method which applies a computation with guaranteed accuracy using rational numbers to design of a control system, and solve $\mathcal{H}_2$ design problem for a single input single output control plant was proposed(Kanno and Hara (2007)). However, there are problems about a method using rational numbers shown below.

(1) It cannot support irrational numbers.
(2) There are inexcusable arithmetic operations since the result of an arithmetic must be closed under an arithmetic by rational numbers.
(3) There are issues about a computational speed and a memory size since an arithmetic of rational numbers may explode.

The purpose of this research is apply a numerical computation with guaranteed accuracy based on floating point numbers to a computation about design of a control system, and design a control system from the viewpoint of a guarantee of a numerical quality. We guaranteed the quality about a computation of design of control system, especially a numerical result about LQ control problem.

In Section 2, the principle of a numerical computation with guaranteed accuracy, and a creation of an interval by the switching the CPU rounding mode to compute a fast numerical computation with guaranteed accuracy using floating point numbers is explained. In Section 3, we propose a LQ control problem based on a numerical computation with guaranteed accuracy. In Section 4, JCGA(Java Computing Guaranteed Accuracy) which is the developed Java numerical computation with guaranteed accuracy package is described. In Section 5, an example of LQ control problem based on numerical computation with guaranteed accuracy using JCGA is explained. Finally, in Section 6, conclusions are described.

2. NUMERICAL COMPUTATION WITH GUARANTEED ACCURACY

2.1 Principle of numerical computation with guaranteed accuracy

A general form of the principle and the method to solve a problem with guaranteed is explained here(Nakao and Yamamoto (1998)).

Now, a problem is given as

$$\text{Find } x \text{ which satisfies } f(x) = 0$$

If an accuracy of an approximation is good, it is hoped that the true solution exists near by the approximate solution of the above problem obtained by a numerical computation. Then, consider a method to check whether the set(candidate set) which contains the approximate solution contains the true solution. If this method is obtained, it is possible to try to specify a set which contains the true solution by the procedure like following.

(1) By using some method, resolve a candidate set which is expected that it contains the true solution .
(2) Verify if the candidate set actually contains the true solution.

(3) If it doesn't contain the true solution, create other candidate set and go back to the previous step.

If a set is specified, the size of the set gives the accuracy of the approximate solution.

The most popular mathematical tool to check whether a set contains a solution of the problem $f(x) = 0$ is an equivalent theorem about an existence of the solution of fixed point equation $x = F(x)$, that is fixed point theorem. There are several types about fixed point theorem, so a generic representation of fixed point theorem is shown below.

Theorem 1. Consider a F(U) as

$$F(U) = \{v \mid v = F(u), u \in U\}$$

where U is a set which satisfies some condition. Then, if set U satisfies

$$F(U) \in U$$

or, U satisfies a similar condition, the true solution of fixed point equation $x = F(x)$ exists within the U (especially exists within the $F(U)$).

2.2 Fast creating of interval by switching CPU rounding mode

A method to search a candidate set by the switching CPU rounding mode is explained here(Oishi (2000a)).

The CPU rounding mode which is called round to nearest(rounds to the floating point number that is the nearest to the real number r) is usually used when we compute using a computer. Other rounding modes are round downward(round towards the floating point number that is the biggest number fewer than the real number r), and round upward(round towards the floating point number that is the smallest number larger than the real number r) etc(ANSI/IEEE (1985)).

In general, it is impossible to solve the true solution itself in a numerical computation using floating point numbers. Then, a numerical computation with guaranteed accuracy switches the CPU rounding mode, computes the infimum and the supremum of the true solution, and creates the interval(G.I.Hargreaves (2002); Oishi (2000a)) as shown in Fig. 1. And, it wraps around the true solution by the interval which has the floating point number as both ends. By using this method, it is possible to search the interval which contains the true solution. Numerical computation with guaranteed accuracy is executed by the simple idea that an interval wraps around a true solution of a problem using an interval.

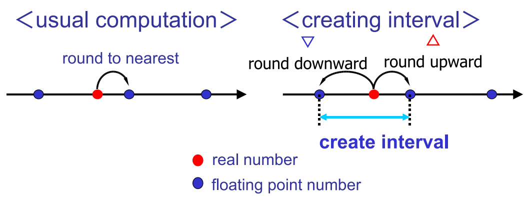

Fig. 1. Difference between usual computation and numerical computation with guaranteed accuracy

3. LQ CONTROL PROBLEM BASED ON NUMERICAL COMPUTATION WITH GUARANTEED ACCURACY

3.1 Verified LQ control problem

This paper proposes verified LQ control problem which verifies numerical result of LQ control problem(Brogan (1981)). It is the example that apply numerical computation with guaranteed accuracy to design of control system and guarantee quality of numerical result.

In LQ control problem, consider the next quadratic form evaluation function (scalar quantity)

$$J = \int_0^\infty [x^T Q x + u^T R u] dt$$

for controllable multi input multi output system (A, B)

$$\dot{x} = Ax + Bu : A(n \times n), B(n \times m) \quad (1)$$

Here, $Q(n \times n)$, $R(m \times n)$ are weighting matrixes, Q is the positive semi definite symmetric matrix, and R is the positive definite symmetric matrix. Then, the optimal feedback control input u which minimizes J is expressed by using feedback coefficient matrix K,

$$u^0 = -Kx \quad (2)$$
$$= -R^{-1}B^T P x \quad (3)$$

Here, $K = R^{-1}B^T P$. And, P is arbitrary $(n \times n)$ symmetric and positive definite matrix, and it is the unique positive definite solution of Riccati matrix equation.

$$A^T P + PA + Q - PBR^{-1}B^T P = 0 \quad (4)$$

Verified LQ control problem is the problem that finds the solution of Riccati equation P with guarantee of numerical quality when solving LQ control problem. The solution method of verified LQ control problem is finding the solution of Riccati equation P which satisfies below conditions.

- $P > 0$
- P is a symmetric matrix
- $[f(P)] := [A^T P + PA + Q - PBR^{-1}B^T P] \supseteq 0$

If P satisfies above conditions, that P is called the verified solution of Riccati equation. And, if P doesn't satisfies conditions, that P is called the rejected solution of Riccati equation.

The third condition of the verification is explained by using Fig. 2. $f(P) := A^T P + PA + Q - PBR^{-1}B^T P$ is the quadratic expression about P by considering variable numbers of $f(P)$ as scalar. Then, drawing the graph of $f(P)$ about P, it is shown like Fig. 2 if it assumes that $A > 0$.

The result of substituting the obtained P to $f(P)$ and computing numerical computation with guaranteed accuracy is obtained by an interval. At this time, by computing using the P which satisfies (4) then, the interval of the result which contains 0 are obtained. Therefore, the interval of computational result is across the positive area and negative area shown like Fig. 2, and it wraps around 0. And, the interval of the computational result exits in only

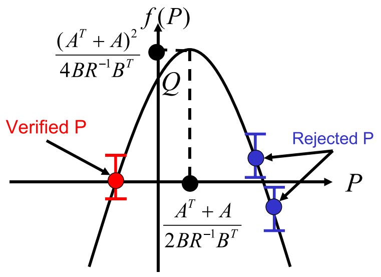

Fig. 2. Verification about P

the positive area or the negative area, the interval doesn't contain 0, and it is find that P doesn't satisfy (4).

3.2 Verified numerical optimal LQ control problem

In the previous subsection, the method that verifies the obtained solution of Riccati equation was shown. After executing verified LQ control problem, if the obtained solution of Riccati equation is a solution of rejected Riccati equation, how to find a verified solution of Riccati equation? In this subsection, a method to find a verified solution of Riccati equation, that is verified numerical optimal LQ control problem is described.

Verified numerical optimal LQ control problem is the problem that pulls out a set of verified solution of Riccati equation(it has n solution) from the interval which contains the true value of the solution, and finds the solution which minimizes a value of evaluation function about the miss match of the design specification.

The control plant is single input single output system given as

$$G(S) = \frac{b_{n-1}s^{n-1} + \cdots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0}$$

and weighting matrixes Q, R are given. Then, verified numerical optimal LQ control problem finds the verified solution of Riccati equation $\tilde{P}$ which makes the distance from the interval

$$[\tilde{Q}] := -A^T\tilde{P} - \tilde{P}A + \tilde{P}BR^{-1}B^T\tilde{P} \qquad (5)$$

minimize. And it makes the distance of the given weighting matrix Q

$$[error_{ij}] = \left[\max_{1\leqq i\leqq n}\max_{1\leqq j\leqq n}(|Q_{ij} - \text{Mid}([\tilde{Q}_{ij}])| + \text{Rad}([\tilde{Q}_{ij}]))\right] \qquad (6)$$

be the evaluation function. Where, n is the order of Q, Mid($[\tilde{Q}_{ij}]$) is the (i,j)th element of the center of $[\tilde{Q}]$, Rad($[\tilde{Q}_{ij}]$) is the (i,j)th element of the radius of $[\tilde{Q}]$. The schematic of the evaluation formula is shown like Fig. 3. Since, $\tilde{Q}$ is a complex number, $[\tilde{Q}]$ is expressed by a circular disk. This evaluation formula computes the sum of the distance from the center of $[\tilde{Q}]$ to the specified weighting matrix Q and the radius of $[\tilde{Q}]$. The true solution of Riccati equation makes $\tilde{Q}$ is completely correspond with weighting matrix Q, so if this value is small, it is thought of as the good solution is obtained. And, if the element of the weighting matrix is not 0, the evaluation value is normalized by dividing by the weighting matrix.

The creation of the candidate set is executed by selecting the floating point number which is in the interval which contains true solution of Riccati equation. At this time,

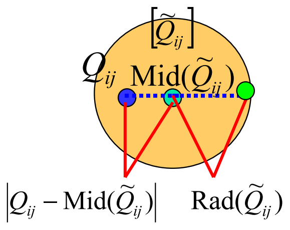

Fig. 3. Schematic of evaluation formula for verified numerical optimal LQ control problem

select the candidate as $\tilde{P}$ must be symmetric. For example, consider

$$P = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}$$

At this time, from the symmetric property of P, $p_{12} = p_{21}$. So when creating the candidate set, pull out the floating point which will be the candidate from the interval of the product set of the interval of (1,2)th element and the interval of (2,1)th element, and make that floating point number be the value of (1,2)th element and (2,1)th element of $\tilde{P}$. By using the above method, candidates which are symmetric matrixes are created.

A solve procedures of verified optimal LQ control problem is shown below.

Solve procedures of verified numerical optimal LQ control problem

Step1 Compute the interval which contains the true value of the solution of Riccati equation, and pull out a candidate set of the verified solution of Riccati equation from it.
Step2 Solve verified LQ control problem
Step3 If it is a verified solution of Riccati equation in Step 2, compute the evaluated value by the evaluation formula(6).
Step4 Make $\tilde{P}$ that makes supremum of the evaluated value the smallest be the verified numerical optimal solution of Riccati equation.

3.3 Methods to solve verified numerical optimal LQ control problem efficiently

When solving Riccati equation with guaranteed accuracy, if apply an algorithm of usual approximate computation(Brogan (1981)) to numerical computation with guaranteed accuracy, it may obtain an interval matrix which has a big radius. If an interval matrix which has big radius is obtained, it increases candidates of verified numerical optimal controller. So, amounts of calculations to select the verified numerical optimal controller is increased. Then, this paper proposes two methods to solve verified numerical optimal LQ control problem efficiently.

Creating a candidate set by using GA One method creates a candidate set by using GA(Hromkovic (2001)) from the interval which contains the true value of Riccati equation(Yano and Koga (2007)).

This method makes a floating point in the interval solution of Riccati equation as individual, use the (6) for the computation of fitness, use UNDX method for the crossover, and use a method by uniform random number for mutation. It is possible to search a candidate set from an interval solution of Riccati equation quickly by using GA.

Solution method of Riccati equation with guaranteed accuracy Another method is a solution method of Riccati equation with guaranteed accuracy. It reduces the radius of interval matrix of solution of Riccati equation obtained by numerical computation with guaranteed accuracy by using Krawczyk method(Oishi (2000a); Nakao and Yamamoto (1998)).

This method defines Riccati equation as non linear equation, and apply Krawczyk method to it and obtained solution of Riccati equation P as an initial interval. It is able to obtain an small radius interval matrix which contains true solution of Riccati equation by using the proposed method. And, differential value which is needed to solve non linear equation is computed by using automatic differentiation(Oishi (2000a)).

4. IMPLEMENTATION OF THE PROPOSED METHOD

4.1 Numerical computation with Guaranteed Accuracy package JCGA

We developed JCGA(Java Computing Guaranteed Accuracy) as a Java package of a numerical computation with guaranteed accuracy. JCGA is able to compute interval arithmetic, interval matrix arithmetic(Oishi (2000a); G.I.Hargreaves (2002)), linear equation(Oishi (1999); G.I.Hargreaves (2002)), eigen value problem(Rump (2001)), polynomial(Oishi (1999)), differential value(Oishi (2000a, 1999, 2000b)), and nonlinear equation(Oishi (2000a); G.I.Hargreaves (2002); Oishi (2000b)) of numerical computation with guaranteed accuracy. And, JCGA use NFC(Koga and Matsuki (2003)) which is a Java numerical computation package for basic operations like matrix operation, or complex number operation etc. Java only use round to nearest mode(Lindholm and Yellin (1999)), and cannot control the CPU rounding mode directly. So, JCGA controls the CPU rounding mode by the calling code of the rounding control written in C language by using JNI(Gordon (1998)).

4.2 Architecture of JCGA

The architecture of JCGA consists of JCGA component which is the main component and executes a computation with guaranteed accuracy, NFC component which computes internal basic operations of JCGA, and libFPUNative.so(for Linux, or FPUNative.dll for Windows) component which controls the CPU rounding mode.

JCGA has round package to switch the CPU rounding mode, interval package to carry out interval arithmetic and interval matrix arithmetic, linear, eigen, polynomial, derivative, nonlinear package to compute each problem with guaranteed accuracy, and pset(Koga and Takei (2002); Koga et al. (2003)) package to manage a processor on a multiprocessor environment. pset package is used to restrain a switch of CPU rounding mode by other process during a computation with guaranteed accuracy.

Classes shown in Table 1 are prepared as main classes to compute with guaranteed accuracy.

Table 1. Classes of numerical computation with guaranteed accuracy

Numerical computation with guaranteed accuracy	Class
Interval arithmetic	Interval
Interval matrix arithmetic	IntervalMatrix
Linear equation	LinearVerifier
Eigen value problem	EigenVerifier
Polynomial	HornerVerifier
Differential value	IntervalDerivative
Nonlinear equation	NonLinearVerifier

4.3 Numerical example using JCGA

This section shows a numerical computation with guaranteed accuracy of eigen value as a numerical example using JCGA.

JCGA provides EigenVerifier class to execute a numerical computation with guaranteed accuracy of eigen value. By using this class, computes guaranteed accuracy of eingen value of the following matrix.

$$A = \begin{bmatrix} -4 & 17 & 60 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

The true value of eigen value is $\lambda = 4, -3, -5$.

The program and the computation result are shown below.

Example program of eigen value with guaranteed accuracy

```
public class EigenVerifierSample {
  public static void main(String[] args) {
    //Matrix A
    Matrix A = new DoubleMatrix(new double[][]{
      {-4, 17, 60},
      { 1,  0,  0},
      { 0,  1,  0}});
    //Creates a numerical computation with
    //guaranteed accuracy object
    //of eigen value
    EigenVerifier ev = new EigenVerifier();
    //Computes eigen value
    //with guaranteed accuracy
    IntervalMatrix ans = ev.solve(A)[0];
    //Display of the center and the radius
    ans.printMidRad("Eigen value");
  }
}
```

Result of eigen value with guaranteed accuracy

```
=== Center of Eigen value(3 x 1) CoMatrix ===
        [ (  1)-Real ]
(  1) 4.00000000000000000E+00
(  2)-3.00000000000000400E+00
(  3)-5.00000000000000000E+00
        [ (  1)-Imag ]
(  1) 0.00000000000000000E+00
(  2) 0.00000000000000000E+00
(  3) 0.00000000000000000E+00
=== Radius of Eigen value(3 x 1) Matrix ===
            (  1)
(  1)2.329383478607396300E-16
(  2)1.437330003186217900E-15
(  3)1.258978188088645200E-15
```

The above result shows that true solution exist within the circular disk which consists of the radius and the center.

And, the existence interval of the solution is computed by using the above result.

Existence interval of eigen value

```
=== Infimum (3 x 1) Matrix ===
            (  1)
 (  1)  3.999999999999999600E+00
 (  2) -3.000000000000002000E+00
 (  3) -5.000000000000002000E+00
=== Supremum (3 x 1) Matrix ===
            (  1)
 (  1)  4.000000000000001000E+00
 (  2) -2.999999999999998700E+00
 (  3) -4.999999999999998000E+00
```

It is confirmed that the true solution is wrapped around between the infimum and the supremum by this result, so the guaranteed accuracy result is validate.

5. NUMERICAL EXAMPLE OF VERIFIED LQ CONTROL PROBLEM

Consider the next problem as an numerical example.

$$G(s) = \frac{1}{s^2+3s+2} = \left[\begin{array}{cc|c} 0 & 1 & 0 \\ -2 & -3 & 1 \\ \hline 1 & 0 & 0 \end{array}\right]$$

and weighting matrixes are

$$Q = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}, \quad R = 1$$

Find a verified solution of Riccati equation of this problem. This problem is able to solve analytically, and the true value is

$$P = \begin{bmatrix} 6 & 1 \\ 1 & 1 \end{bmatrix}, \quad K = [\,1\ 1\,]$$

First, computes the solution of Riccati equation and feedback gain by Matlab(Mat). The version of Matlab is 7.4.0(R2007a), and the function care is used.

```
P =
    6.000000000000002e+000   9.999999999999996e-001
    9.999999999999996e-001   1.000000000000000e+000

K =
    9.999999999999996e-001    1.000000000000000e+000
```

Substitute this P to $f(P)$, and computes numerical computation with guaranteed accuracy by using JCGA

```
===  Infimum of f(P)  (   2  x   2)  Matrix  ===
           (  1)            (  2)
(  1)  2.553512956637860000E-15  3.552713678800501000E-15
(  2)  3.552713678800501000E-15 -8.881784197001252000E-16
===  Supremum of f(P)  (   2  x   2)  Matrix  ===
           (  1)            (  2)
           (  1)            (  2)
 (  1)  2.664535259100375700E-15  3.552713678800501000E-15
 (  2)  3.552713678800501000E-15 -8.881784197001252000E-16
```

The above result shows that it doesn't wrap around 0. So, this P is the rejected solution of Riccati equation.

Then, solve the verified numerical optimal LQ control problem about this problem.

Step1 : Computes the solution of Riccati equation, and that value is

Interval of solution of Riccati equation

```
===  Infimum of IP  (   2  x   2)  Matrix  ===
           (  1)            (  2)
(  1)  5.999999999999988000E+00  9.999999999999917000E-01
(  2)  9.999999999999970000E-01  9.999999999999976000E-01
===  Supremum of IP  (   2  x   2)  CoMatrix  ===
           (  1)            (  2)
(  1)  6.000000000000011000E+00  1.000000000000007300E+00
(  2)  1.000000000000002400E+00  1.000000000000002200E+00
```

then, it finds that this interval contains the true solution of Riccati equation. At this moment, pull out all of floating points from the inside this interval, verify the solution of Riccati equation P.

(1,1)th element of the obtained interval has 27 floating points, $(1,2)$th element has 109 floating points, $(2,1)$th element has 109 floating points, and $(2,2)$th element has 33 floating points. However, considering from the viewpoint of interval, $(1,2)$th element contains $(2,1)$th element. So, pull out the floating point of the candidate from only $(2,1)$th element which is the product set of $(1,2)$th element and $(2,1)$th element. Therefore, quantity of candidate set is $27 \times 109 \times 33 = 97119$.

Then, it reduces the radius of this interval matrix by using Krawczyk method.

Interval after using Krawczyk method

```
===  Infimum of IP  (   2  x   2)  Matrix  ===
           (  1)            (  2)
 (  1)  5.999999999999999000E+00  9.999999999999999000E-01
 (  2)  9.999999999999999000E-01  9.999999999999999000E-01
 ===  Supremum of IP  (   2  x   2)  Matrix  ===
           (  1)            (  2)
 (  1)  6.000000000000001000E+00  1.000000000000000200E+00
 (  2)  1.000000000000000200E+00  1.000000000000000700E+00
```

(1,1)th element of the obtained interval has 3 floating points, $(1,2)$th element has 3 floating points, $(2,1)$th element has 3 floating points, and $(2,2)$th element has 5 floating points. So, quantity of candidate set is $3 \times 3 \times 9 = 45$. And, the radius of this interval is reduced with the interval contains true solution. So, it is able to reduce the radius of the interval matrix and the quantity of the candidate set by using Krawczyk method.

Step2 : There are 1 verified solution of Riccati equation.

Step3:We compute the evaluated value about the solution of Riccati equation based on the evaluation formula.

Step4 : That value is the below solution.

Verified numerical optimal solution of Riccati equation

```
===  P  (   2  x   2)  Matrix  ===
           (  1)            (  2)
(  1)  6.000000000000000000E+00 1.000000000000000000E+00
(  2)  1.000000000000000000E+00 1.000000000000000000E+00
```

And, the feedback gain obtained by using the above value is

Numerical optimal feedback gain

```
=== K  (   1  x   2)  Matrix  ===
            ( 1)          ( 2)
( 1)  1.000000000000000000E+00  1.000000000000000000E+00
```

The above value is correspond with the analytical solution. So, the evaluated value of the above result is

$$error = [0.0, 0.0]$$

This is because, values 1 and 6 which are appeared as the element of P are able to express the format of IEEE754(ANSI/IEEE (1985)) which is the specification of the floating point number shown as below

$$(-1)^s 2^{e-1023}(1.f)$$

and, it is possible to express the floating point number.

Substitute the above P to $f(P)$,

```
===  Infimum of f(P)  (   2  x   2)  Matrix  ===
            ( 1)          ( 2)
( 1)                        0                          0
( 2)                        0                          0
===  Supremum of f(P)  (   2  x   2)  Matrix  ===
            ( 1)          ( 2)
( 1)                        0                          0
( 2)                        0                          0
```

and, it is correspond with 0.

The time to solve this problem is shown as Table 2. Computation environment is OS: Windows XP Professional, CPU: Pentium D 945(3.4GHz), Memory: 1024MB.

Table 2. Time to compute the example

Step	Time(ms)
Step1(Computation of solution of Riccati equation + select candidates)	540.5(535.8+4.7)
Step2(Verified LQ control problem)	42.1
Step3(Evaluation of verified solution of Riccati equation)	0
Total	582.6

The time to solve this problem without using Krawczyk method is shown as Table 3. By using Krawczyk method, it is able to compute about 1/8 times. So, the usefulness of the proposed method is shown.

Table 3. Time to compute the example without using Krawczyk method

Step	Time(ms)
Step1(Computation of solution of Riccati equation + select candidates)	659.4(54.7+604.7)
Step2(Verified LQ control problem)	40092.1
Step3(Evaluation of verified solution of Riccati equation)	0
Total	40751.5

6. CONCLUSIONS

This paper developed JCGA which is a Java numerical computation with guaranteed accuracy package, and applied numerical computation with guaranteed accuracy to computational problems about design of control system, especially guarantee a quality about LQ control problem. It is possible to guarantee the numerical quality about design of control of system. We would like to go on to propose design methods for another design problems like robust control etc. from the viewpoint of the guarantee of numerical quality.

REFERENCES

The MathWorks Matlab. http://www.mathworks.com/products/matlab/.

ANSI/IEEE. IEEE Standard for Binary Floating Point Arithmetic, 1985. Std 754-1985 edition, New York.

William L. Brogan. *Modern Control Theory Third Edition.* Prentice-Hall International, Inc., 1981.

G.I.Hargreaves. *Interval Analysis in MATLAB*, 2002. Numerical Analysis Report No.416.

Rob Gordon. *Essential JNI: Java Native Interface.* Prentice Hall Ptr, 1998.

Juraj Hromkovic. *Algorithmics for Hard Computing Problems:Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics.* Springer, 2001.

Masaaki Kanno and Shinji Hara. Guaranteed Accuracy Algorithm in H2 Optimal Tracking Controller Synthesis. *Transactions of the Society of Instrument and Control Engineers*, 43(2):102–109, 2007.

Masanobu Koga and Takeshi Matsuki. Development of OS-Neutral Numerical Foundation Class Library and its Application to Control System, 2003. SICE 3th Annual Conference on Control Systems pp.725-728.

Masanobu Koga and Kiminori Takei. Parallel Computation guaranteed Accuracy using Processor Set, 2002. 21th Simulation Technology Conference, pp.199-202.

Masanobu Koga, Satoko Tsuneoka, and Akinori Hiroki. Parallel Computing Guaranteed Accuracy using Java, 2003. FIT2003 3rd Forum on Information Technology, pp.37-39.

Tim Lindholm and Frank Yellin. *The Java Virtual Machine Specification, Second Edition.* Addison-Wesley Pub, 1999.

Mitsuhiro Nakao and Nobito Yamamoto. *Numerical Computation with Guaranteed Accuracy.* NIPPON HYORONSHA CO.,LTD.PURBLISHERS, 1998.

Shinichi Oishi. *Numerical Calculation.* SHOKABO PUBLISHING Co., Ltd., 1999.

Shinichi Oishi. *Verified Numerical Computation.* CORONA PUBLISHING CO., LTD., 2000a.

Shinichi Oishi. *Tools for Numerical Computation on Linux.* CORONA PUBLISHING CO., LTD., 2000b.

Siegried M. Rump. Computational error bounds for multiple or nearly multiple eigenvalues. *Linear Algebra and its Applications*, 324:209–226, 2001.

Kentaro Yano and Masanobu Koga. Pole assignment method based on numerical computation with guaranteed accuracy, 2007. SICE Annual Conference 2007.