数 値 解 析 27－1
（１９８８．１２．２）

精度保証付き数値計算について

山本　哲朗，　陳　小君，　菅野　幸夫
（愛媛大理）

この報告は前回 [7]における山本の報告の続きであって，菅野が作成した区間演算ソフトを用いて行った数値実験の結果を中心に述べる．現在区間解析に関する論理的考察は多くの人々によりなされているが，小規模な問題に対しても所要計算時間を含む具体的な資料は殆ど公表されていないように見えるから，この報告が今後この方面に関心を抱く人達のために少しでも役立つことを期待したい．また，最後に中国における区間解析の現状を，最近出版された書物 [6]に基づき，陳が報告する．

ON NUMERICAL COMPUTATION WITH VALIDATION

Tetsuo YAMAMOTO, Xiaonjun CHEN, Sachio KANNO

Department of Mathematics
Faculty of Science, Ehime University
Matsuyama 790, Japan

This is a continuation of a previous report [7] given by Yamamoto. Some results of numerical compuation will be reported, which were obtained by an interval arithmetic package programmed by Kanno. Although much literature has been published on interval mathematics, it seems to the authors that little mention has been made on the results of numerical experiment including computing time. Therefore we expect that this report will be useful to those who are interested in interval arithmetic. Finally Chen will report the recent state of the study on interval mathematics in China on the basis of a recent book [6] by Wang, Zhang and Deng.

§ 1. はじめに

出力された数値に精度保証（品質保証）を与える問題は数値解析における主題の一つであり，古来多くの研究がある．現代における主要な方法は，次の４種であろう．

i) 区間解析法（Moore, Nickel, Rall,奥田・棚町, 等），（IBM 社 ACRITH ソフトもこれに含まれる．）

ii) 剰余演算法（Gregory-Krishnamurthy[3]等）

iii) 数式処理（佐々木, 野田, 等）

iv) 事後誤差評価法と区間解析法の融合

現在のところ，これらの方法は計算時間と簡便さの点で，大型問題には使えないが，将来電子計算機の発達とともに，次第に実用化されていくことを期待したい．本講演では，区間解析法による精度保証付き数値計算につき，菅野が作成したソフトを用いて数値実験を行った結果を中心に報告する．

§ 2. 区間解析法

この方法における基本概念と基本定理（Krawczyk-Moore）および関数方程式への適用例についてはすでに本研究会において山本が Moore[5] および Kulisch-Miranker[4] に基づき報告した [7]．また，偏微分方程式に対する解の存在を電子計算機の上で自動的に検証しようとする試みが中尾（九大理）によりなされている．今回は，[7] の続報として，中国における研究状況を最近の書物 王・張・[illegible] [6]に基づき陳が報告する．近年における欧米の研究動向については，昨年 Columbus(Ohio)で開催された国際研究会の模様を中心にして中尾充宏氏が別稿で報告される予定である．

§ 3. 区間演算による精度保証の試み（実験報告）

3. 1 計算に用いたシステム

精度保証付き演算をおこなうため，任意多倍長のポイント計算，保証付きインターバル計算のソフトを作成した．任意多倍長精度保証付きインターバル計算の概略は，以下のようなものである．

計算の実行に伴う保証区間の広がりを抑えるため，アキュムレータ変数を用意した．アキュムレータ変数とは，仮数部のビット長が一般の変数より長い変数である．通常ｎ倍精度の計算において，アキュムレータ変数は，32(n+1) ビット，一般変数は，32n ビットの仮数部がとられる．

ポイントの四則演算，addrp, subrp, mulrp, divrp はアキュムレータ変数に対して，一般変数の演算を施すことによって為される．このときオプションの指定によって，次の操作が可能である．

1) 完全な演算結果が仮数部 32(n+1)ビットに納まらないとき，そのすぐ下の値を返す．(opt=-1)

2) 完全な演算結果が仮数部 32(n+1)ビットに納まらないとき，第 32(n+1)+1ビットの値によって零捨一入をおこなう．(opt= 0)

3) 完全な演算結果が仮数部 32(n+1)ビットに納まらないとき，そのすぐ上の値を返す．(opt= 1)

インターバルの四則演算，addri, subri, mulri, divri は，オプション指

定をしたポイントの四則演算によりインターバルの上限と下限を計算している．実際の組合せを 表１ に示す．

addrp, subrp, mulrp, divrp, addri, subri, mulri, divri の１秒間の演算実行回数を 表２に示す．参考までに示すが，使用した計算機（Apollo DOMAIN 3000）にインプリメントされている浮動小数点演算での値は，単精度，倍精度とも 25000回／秒であった．

３．２ 基本関数の保証付き計算

無限級数 $\sum_{n=0}^{\infty} a_n$ は収束するとして

$$s=\sum_{n=0}^{\infty} a_n$$

とする．$s_n=\sum_{i=0}^{n} a_i$ とおくならば，

$$s= s_n + R_n \quad （但し\ R_n=\sum_{i=n+1}^{\infty} a_n \quad ）$$

今，保証付きインターバル計算によって，アキュムレータ変数 $\tilde{a}cc$ に各項 a_n を足し込んでゆき，Ｎ項まで足し込んだ時点で $\tilde{a}cc$ を上下方向に広げてやるとする．（操作１）

このとき，

$$^{\exists}N_0\ ,\ n\geq N_0 \quad ならば \quad |a_{n+1}|\leq \frac{1}{2}|a_n| \qquad （１）$$

が満たされるならば，

$$|R_{n_0}|\leq|a_{n_0}|$$

$N\geq N_0$ とする（（１）が満たされるまでは，足し込みを続けるとする）．

$$|R_N|<|a_N|\leq 2^{optmin(\tilde{a}cc)-1-1}$$

よって

$$upbnd(abs(a_N))\leq 2^{optmin(\tilde{a}cc)-1-1},$$

が満たされるならば，操作１の後の $\tilde{a}cc$ の値は保証される．

$$2^{optmax(a_N)-1}\leq upbnd(abs(a_N))<2^{optmax(a_N)}$$

故に，条件（１）が満足された後の足し込みの停止条件としては，

$$2^{\mathrm{optmax}(a_N)} \leq 2^{\mathrm{optmax}(\tilde{a}cc)-l-1}$$

すなわち

$$\mathrm{optmin}(acc)-\mathrm{optmax}(a_N) \geq l+1$$

を用いることが出来る．
よって次のループが考えられる．

```
ãcc= [0,0]
n= -1
eps= 0
DO WHILE (n<N0 OR eps<l+1)
  n= n+1
  COMPUTE an
  ãcc= ãcc + an
  eps= optmin(ãcc) - optmax(an)
END DO
ext(ãcc)
```

rem. l はアキュムレータ変数の仮数部のビット長を表す．
指数の底は，２である．
abs() はインターバルの絶対値を表す．
upbnd() はインターバルの上限を表す．
optmin(),optmax() はそれぞれ，インターバル変数の上限，下限の指数部の小さいほう，大きいほうを表す．

３．３　計算例１

行列 $A = \begin{pmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{pmatrix}$ に対する逆行列 $A^{-1} = \begin{pmatrix} 68 & -41 & -17 & 10 \\ -41 & -25 & 10 & -6 \\ -17 & 10 & 5 & -3 \\ 10 & -6 & -3 & 2 \end{pmatrix}$，

積 $A \cdot A^{-1}$ のポイント計算，インターバル計算の結果を示す．逆行列の計算は，部分ピボッティング・ガウスの消去法を用いた．（[1] Chap.15, [2] Example 3.5）

```
CPU TIME: 0:00:00.58
L(  1,  1)
pt. +0.6799999982 *10^2
pt. +0.6799999983 *10^2
L(  1,  2)
pt. -0.4099999987 *10^2
pt. -0.4099999986 *10^2
L(  1,  3)
pt. -0.1699999994 *10^2
pt. -0.1699999993 *10^2
L(  1,  4)
pt. +0.9999999959 *10^1
pt. +0.9999999960 *10^1
L(  2,  1)
pt. -0.4099999990 *10^2
pt. -0.4099999989 *10^2
L(  2,  2)
pt. +0.2499999992 *10^2
pt. +0.2499999993 *10^2
L(  2,  3)
pt. +0.9999999962 *10^1
pt. +0.9999999963 *10^1
L(  2,  4)
pt. -0.5999999976 *10^1
pt. -0.5999999975 *10^1
L(  3,  1)
pt. -0.1699999995 *10^2
pt. -0.1699999994 *10^2
L(  3,  2)
pt. +0.9999999970 *10^1
pt. +0.9999999971 *10^1
L(  3,  3)
pt. +0.4999999985 *10^1
pt. +0.4999999986 *10^1
L(  3,  4)
pt. -0.2999999991 *10^1
pt. -0.2999999990 *10^1
L(  4,  1)
pt. +0.9999999966 *10^1
pt. +0.9999999967 *10^1
L(  4,  2)
pt. -0.5999999982 *10^1
pt. -0.5999999981 *10^1
L(  4,  3)
pt. -0.2999999991 *10^1
pt. -0.2999999990 *10^1
L(  4,  4)
pt. +0.1999999994 *10^1
pt. +0.1999999995 *10^1

CPU TIME: 0:00:01.08
L(  1,  1)
lo. +0.6799999827 *10^2
up. +0.6800000150 *10^2
L(  1,  2)
lo. -0.4100000088 *10^2
up. -0.4099999888 *10^2
L(  1,  3)
lo. -0.1700000044 *10^2
up. -0.1699999950 *10^2
L(  1,  4)
lo. +0.9999999701 *10^1
up. +0.1000000027 *10^2
L(  2,  1)
lo. -0.4100000083 *10^2
up. -0.4099999924 *10^2
L(  2,  2)
lo. +0.2499999954 *10^2
up. +0.2500000038 *10^2
L(  2,  3)
lo. +0.9999999780 *10^1
up. +0.1000000019 *10^2
L(  2,  4)
lo. -0.6000000110 *10^1
up. -0.5999999867 *10^1
L(  3,  1)
lo. -0.1700000031 *10^2
up. -0.1699999962 *10^2
L(  3,  2)
lo. +0.9999999780 *10^1
up. +0.1000000018 *10^2
L(  3,  3)
lo. +0.4999999893 *10^1
up. +0.5000000088 *10^1
L(  3,  4)
lo. -0.3000000055 *10^1
up. -0.2999999934 *10^1
L(  4,  1)
lo. +0.9999999769 *10^1
up. +0.1000000019 *10^2
L(  4,  2)
lo. -0.6000000112 *10^1
up. -0.5999999864 *10^1
L(  4,  3)
lo. -0.3000000055 *10^1
up. -0.2999999933 *10^1
L(  4,  4)
lo. +0.1999999959 *10^1
up. +0.2000000034 *10^1

A * L
C(  1,  1)
pt. +0.1000000074 *10^1
pt. +0.1000000075 *10^1
C(  1,  2)
pt. +0.5215406417 *10^-7
pt. +0.5215406418 *10^-7
C(  1,  3)
pt. +0.2607703208 *10^-7
pt. +0.2607703209 *10^-7
C(  1,  4)
pt. .ZERO.
pt. .ZERO.
C(  2,  1)
pt. .ZERO.
pt. .ZERO.
C(  2,  2)
pt. +0.1000000059 *10^1
pt. +0.1000000060 *10^1
C(  2,  3)
pt. +0.2980232238 *10^-7
pt. +0.2980232239 *10^-7
C(  2,  4)
pt. .ZERO.
pt. .ZERO.
C(  3,  1)
pt. -0.4470348359 *10^-7
pt. -0.4470348358 *10^-7
C(  3,  2)
pt. +0.8940696716 *10^-7
pt. +0.8940696717 *10^-7
C(  3,  3)
pt. +0.1000000044 *10^1
pt. +0.1000000045 *10^1
C(  3,  4)
pt. -0.1117587090 *10^-7
pt. -0.1117587089 *10^-7
C(  4,  1)
pt. +0.5215406417 *10^-7
pt. +0.5215406418 *10^-7
C(  4,  2)
pt. +0.5960464477 *10^-7
pt. +0.5960464478 *10^-7
C(  4,  3)
pt. +0.2980232238 *10^-7
pt. +0.2980232239 *10^-7
C(  4,  4)
pt. +0.9999999981 *10^0
pt. +0.9999999982 *10^0

A * L
C(  1,  1)
lo. +0.9999839514 *10^0
up. +0.1000016004 *10^1
C(  1,  2)
lo. -0.9492039881 *10^-5
up. +0.9447336197 *10^-5
C(  1,  3)
lo. -0.4619359971 *10^-5
up. +0.4597008229 *10^-5
C(  1,  4)
lo. -0.2793987724 *10^-5
up. +0.2801418305 *10^-5
C(  2,  1)
lo. -0.2250075341 *10^-4
up. +0.2238154412 *10^-4
C(  2,  2)
lo. +0.9999867081 *10^0
up. +0.1000013233 *10^1
C(  2,  3)
lo. -0.6459653378 *10^-5
up. +0.6459653378 *10^-5
C(  2,  4)
lo. -0.3907829524 *10^-5
up. +0.3907829524 *10^-5
C(  3,  1)
lo. -0.2051889897 *10^-4
up. +0.2047419549 *10^-4
C(  3,  2)
lo. -0.1212209464 *10^-4
up. +0.1208484173 *10^-4
C(  3,  3)
lo. +0.9999940954 *10^0
up. +0.1000005883 *10^1
C(  3,  4)
lo. -0.3580003977 *10^-5
up. +0.3580003977 *10^-5
C(  4,  1)
lo. -0.1812726260 *10^-4
up. +0.1807510853 *10^-4
C(  4,  2)
lo. -0.1071020961 *10^-4
up. +0.1068923142 *10^-4
C(  4,  3)
lo. -0.5209818483 *10^-5
up. +0.5191192031 *10^-5
C(  4,  4)
lo. +0.9999968390 *10^0
up. +0.1000003165 *10^1
```

3．4　計算例2

連立1次方程式

$$\begin{pmatrix} (1-10^{-n}) & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -3 \\ -2 \\ -1 \\ -3 \end{pmatrix}$$

を 1 倍精度で解いた．誤差のない解は，

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 10^n \\ 10^n + 1 \\ 10^n + 2 \\ 10^n + 3 \end{pmatrix}$$

である．([2] Example 3.21)

```
n= 3                          n= 3                          n= 5                          n= 5
CPU TIME: 0:00:00.22          CPU TIME: 0:00:00.43          CPU TIME: 0:00:00.20          CPU TIME: 0:00:00.43
X( 1)                         X( 1)                         X( 1)                         X( 1)
pt. +0.9999998359 *10^3       lo. +0.9999988985 *10^3       pt. +0.9999923855 *10^5       lo. +0.9998892474 *10^5
pt. +0.9999998360 *10^3       up. +0.1000001007 *10^4       pt. +0.9999923856 *10^5       up. +0.1000108809 *10^6
X( 2)                         X( 2)                         X( 2)                         X( 2)
pt. +0.1000999835 *10^4       lo. +0.1000999601 *10^4       pt. +0.1000002385 *10^6       lo. +0.9999791012 *10^5
pt. +0.1000999836 *10^4       up. +0.1001000304 *10^4       pt. +0.1000002386 *10^6       up. +0.1000048956 *10^6
X( 3)                         X( 3)                         X( 3)                         X( 3)
pt. +0.1001999835 *10^4       lo. +0.1001999601 *10^4       pt. +0.1000012385 *10^6       lo. +0.9999891012 *10^5
pt. +0.1001999836 *10^4       up. +0.1002000304 *10^4       pt. +0.1000012386 *10^6       up. +0.1000058956 *10^6
X( 4)                         X( 4)                         X( 4)                         X( 4)
pt. +0.1002999835 *10^4       lo. +0.1002999601 *10^4       pt. +0.1000022385 *10^6       lo. +0.9999991012 *10^5
pt. +0.1002999836 *10^4       up. +0.1003000304 *10^4       pt. +0.1000022386 *10^6       up. +0.1000068956 *10^6

n= 4                          n= 4                          n= 6                          n= 6
CPU TIME: 0:00:00.20          CPU TIME: 0:00:00.42          CPU TIME: 0:00:00.20          CPU TIME: 0:00:00.42
X( 1)                         X( 1)                         X( 1)                         X( 1)
pt. +0.9999993701 *10^4       lo. +0.9999900508 *10^4       pt. +0.9999923852 *10^6       lo. +0.9990609621 *10^6
pt. +0.9999993702 *10^4       up. +0.1000011019 *10^5       pt. +0.9999923853 *10^6       up. +0.1001158908 *10^7
X( 2)                         X( 2)                         X( 2)                         X( 2)
pt. +0.1000099370 *10^5       lo. +0.1000097040 *10^5       pt. +0.9999933852 *10^6       lo. +0.9997806105 *10^6
pt. +0.1000099371 *10^5       up. +0.1000104030 *10^5       pt. +0.9999933853 *10^6       up. +0.1000459260 *10^7
X( 3)                         X( 3)                         X( 3)                         X( 3)
pt. +0.1000199370 *10^5       lo. +0.1000197040 *10^5       pt. +0.9999943852 *10^6       lo. +0.9997616105 *10^6
pt. +0.1000199371 *10^5       up. +0.1000204030 *10^5       pt. +0.9999943853 *10^6       up. +0.1000460260 *10^7
X( 4)                         X( 4)                         X( 4)                         X( 4)
pt. +0.1000299370 *10^5       lo. +0.1000297040 *10^5       pt. +0.9999953852 *10^6       lo. +0.9997626105 *10^6
pt. +0.1000299371 *10^5       up. +0.1000304030 *10^5       pt. +0.9999953853 *10^6       up. +0.1000461260 *10^7
```

3．3　計算例3

ヒルベルト行列 $A_n = \left(a_{ij}^{(n)} \right)$, $a_{ij}^{(n)} = \frac{1}{i+j-1}$ の逆行列の第 1 列を $A_n x = e_1$ を解くことにより求めた．([2] Example 3.8)

```
n= 8                                  n= 8
CPU TIME: 0:00:03.42                  CPU TIME: 0:00:07.05
L( 1)                                 L( 1)
pt. +0.6400000000 0094653021 *10^2    lo. +0.5708021332 1746681904 *10^2
pt. +0.6400000000 0094653022 *10^2    up. +0.7091978667 8411210153 *10^2
L( 2)                                 L( 2)
pt. -0.2016000000 0056266150 *10^4    lo. -0.2024531044 6020446699 *10^4
pt. -0.2016000000 0056266149 *10^4    up. -0.2007468955 4087318877 *10^4

n= 12                                                 n= 12
CPU TIME: 0:00:18.75                                  CPU TIME: 0:00:38.15
L( 1)                                                 L( 1)
pt. +0.1440000000 0000019508 9517549469 *10^3         lo. -0.3831955860 4824310381 1410174597 *10^7
pt. +0.1440000000 0000019508 9517549470 *10^3         up. +0.3832243860 4824310392 8491393327 *10^7
L( 2)                                                 L( 2)
pt. -0.1029600000 0000025082 4963244119 *10^5         lo. -0.4734457556 1986811188 6160894395 *10^7
pt. -0.1029600000 0000025082 4963244118 *10^5         up. +0.4713865556 1986809684 2763550595 *10^7
```

```
n= 20
CPU TIME: 0:02:12.83
L(  1)
pt. +0.4000000000 0007660252 1914155006 8869483158 *10^3
pt. +0.4000000000 0007660252 1914155006 8869483159 *10^3
L(  2)
pt. -0.7980000000 0028694272 5872652732 8118415348 *10^5
pt. -0.7980000000 0028694272 5872652732 8118415347 *10^5
```

```
n= 12
CPU TIME: 0:00:57.72
L(  1)
lo. +0.1439991035 9823067028 5938970433 3862959980 *10^3
up. +0.1440008964 0176932971 4061124351 6560971129 *10^3
L(  2)
lo. -0.1029600110 5072135840 1120392111 2083092638 *10^5
up. -0.1029599889 4927864159 8879728704 6706021080 *10^5
```

```
n= 20
CPU TIME: 0:08:01.33
L(  1)
lo. -0.5191758232 2878444181 4201896849 3581386536 2075530665 2896631367 *10^12
up. +0.5191758240 2878444181 4201896849 3581386536 3670065271 2821972013 *10^12
L(  2)
lo. -0.6400341404 5647056269 2971067203 5655096398 2605190153 4293696522 *10^12
up. +0.6400339808 5647056269 2971067203 5655096331 6360821126 1923465141 *10^12
```

3．6　計算例4

ヒルベルト行列の逆行列の第1列のインターバル計算に Krawczyk のアルゴリズムを用い，保証区間をせばめることを試みた．([5])

```
n= 12
iteration 4 times.
CPU TIME: 0:05:20.68
L(  1)
lo. +0.1439999999 9999892602 5103672672 *10^3
up. +0.1440000000 0000108990 5291480302 *10^3
L(  2)
lo. -0.1029600000 0000142555 8080455461 *10^5
up. -0.1029599999 9999880138 1867180407 *10^5

n= 16
iteration 3 times.
CPU TIME: 0:20:50.37
L(  1)
lo. +0.2559999999 9999999999 9999999890 6143088369 5403000497 *10^3
up. +0.2560000000 0000000000 0000000074 7328917288 9547080867 *10^3
L(  2)
lo. -0.3264000000 0000000000 0000000162 6102201224 3283312872 *10^5
up. -0.3263999999 9999999999 9999999748 3896976359 1518770704 *10^5

n= 17 ( precision= 5 )
- WARNING! intersection is empty 'insec'
```

```
n= 20
iteration 4 times.
CPU TIME: 1:04:43.18
L(  1)
lo. +0.3999999999 9999999999 9999999999 9606624586 7257096816 4373518710 *10^3
up. +0.4000000000 0000000000 0000000000 0218922049 4489641795 3783541547 *10^3
L(  2)
lo. -0.7980000000 0000000000 0000000000 0843168418 7958504716 0489961762 *10^5
up. -0.7979999999 9999999999 9999999999 8519455506 1706032378 6922838334 *10^5
L( 19)
lo. +0.1343120024 3999999999 9999999999 8921906406 2240400957 7440138857 *10^14
up. +0.1343120024 4000000000 0000000000 0619060987 6943076518 5205670545 *10^14
L( 20)
lo. -0.1378465288 2000000000 0000000000 0637740985 6036452145 2978746424 *10^13
up. -0.1378465288 1999999999 9999999999 8837284571 4394414068 8652343929 *10^13
```

3．7　計算例5

ヒルベルト行列の逆行列の第1列をポイント計算によって求め，文献[8]の方法により近似逆行列を用いて精度保証をおこなった．精度保証はインターバル計算で行っている．

```
n= 12                                  n= 13
CPU TIME: 0:01:13.43                   CPU TIME: 0:01:31.50
||kappa||                              ||kappa||
lo. +0.3814904557 0385897718 *10^-2    lo. +0.7166325139 4278715267 *10^0
up. +0.3862155652 7460576547 *10^-2    up. +0.7172043466 7943377960 *10^0

L(  1)
lo. +0.1423520574 9716779449 *10^3     lo. -0.3854164643 7115562247 *10^5
up. +0.1456500503 7453150816 *10^3     up. +0.3887968382 6341801201 *10^5
L(  2)
lo. -0.1031408841 3883990958 *10^5     lo. -0.4192014350 7362955967 *10^6
up. -0.1027817957 8796403102 *10^5     up. +0.3908039064 6934346987 *10^6

n= 12                                              n= 19
CPU TIME: 0:01:56.58                               CPU TIME: 0:07:17.80
||kappa||                                          ||kappa||
lo. +0.8955836763 4159315631 4792924664 *10^-12    lo. +0.2212086878 7200471593 7623853250 *10^1
up. +0.8956932247 5791914853 2606734816 *10^-12    up. +0.2212086364 8801005248 8506484982 *10^1

L(  1)
lo. +0.1439999999 9999960072 0907349822 *10^3
up. +0.1440000000 0000078945 8127749117 *10^3
L(  2)
lo. -0.1029600000 0000097701 4751071680 *10^5
up. -0.1029599999 9999952423 5175416558 *10^5
```

```
n= 20
CPU TIME: 0:12:36.23
||kappa||
lo. +0.1121788371 3310473679 6618374709 3085053443 *10^-7
up. +0.1121793243 9108838751 1197440498 4510400108 *10^-7

L(  1)
lo. +0.3999999999 9792373583 3116414888 8207913684 *10^3
up. +0.4000000000 0222946921 0711895124 9531052632 *10^3
L(  2)
lo. -0.7980000000 0338244843 4113711640 2417887416 *10^5
up. -0.7979999999 9719143701 7631593825 3818943279 *10^5
```

3．8　計算例6

2点境界値問題 $u''+pu+q=0$, $p(x)=-\sin(2\pi x)\cdot\exp(x)$, $q(x)=1$, 境界条件 $u(0)=u(1)=0$ に対する中心差分近似解の計算結果を示す.

```
division: 512              division: 512
CPU TIME: 0:04:58.65       CPU TIME: 0:09:53.20
CPU TIME: 0:00:19.35       CPU TIME: 0:00:39.28

X(  1)                     X(  1)
pt. +0.9562689638 *10^-3   lo. +0.9562476727 *10^-3
pt. +0.9562689639 *10^-3   up. +0.9562740948 *10^-3
X(  2)                     X(  2)
pt. +0.1908723274 *10^-2   lo. +0.1908680693 *10^-2
pt. +0.1908723275 *10^-2   up. +0.1908733537 *10^-2
X(  3)                     X(  3)
pt. +0.2857363067 *10^-2   lo. +0.2857299197 *10^-2
pt. +0.2857363068 *10^-2   up. +0.2857378458 *10^-2

X(256)                     X(256)
pt. +0.1278966206 *10^0    lo. +0.1278933013 *10^0
pt. +0.1278966207 *10^0    up. +0.1278974541 *10^0
X(257)                     X(257)
pt. +0.1279485251 *10^0    lo. +0.1279452074 *10^0
pt. +0.1279485252 *10^0    up. +0.1279493589 *10^0
X(258)                     X(258)
pt. +0.1279966051 *10^0    lo. +0.1279932889 *10^0
pt. +0.1279966052 *10^0    up. +0.1279974391 *10^0

X(509)                     X(509)
pt. +0.3109100001 *10^-2   lo. +0.3109039332 *10^-2
pt. +0.3109100002 *10^-2   up. +0.3109116544 *10^-2
X(510)                     X(510)
pt. +0.2076548426 *10^-2   lo. +0.2076507978 *10^-2
pt. +0.2076548427 *10^-2   up. +0.2076559456 *10^-2
X(511)                     X(511)
pt. +0.1040181628 *10^-2   lo. +0.1040161404 *10^-2
pt. +0.1040181629 *10^-2   up. +0.1040187143 *10^-2
```

```
addri(acc,ord)
  acc.lo= acc.lo + ord.lo (opt=-1)
  acc.up= acc.up + ord.up (opt= 1)

subri(acc,ord)
  acc.lo= acc.lo - ord.up (opt=-1)
  acc.up= acc.up - ord.lo (opt= 1)

divri(acc,ord)
(case ++,++)
  acc.lo= acc.lo / ord.up (opt=-1)
  acc.up= acc.up / ord.lo (opt= 1)
(case ++,--)
  acc.lo= acc.up / ord.up (opt=-1)
  acc.up= acc.lo / ord.lo (opt= 1)
(case -+,++)
  acc.lo= acc.lo / ord.lo (opt=-1)
  acc.up= acc.up / ord.lo (opt= 1)
(case -+,--)
  acc.lo= acc.up / ord.up (opt=-1)
  acc.up= acc.lo / ord.up (opt= 1)
(case --,++)
  acc.lo= acc.lo / ord.lo (opt=-1)
  acc.up= acc.up / ord.up (opt= 1)
(case --,--)
  acc.lo= acc.up / ord.lo (opt=-1)
  acc.up= acc.lo / ord.up (opt= 1)

rem. --: upbnd < 0, ++: 0 < lobnd,
     -+: otherwise.
```

```
mulri(acc,ord)
(case ++,--)
  acc.lo= acc.lo * ord.lo (opt=-1)
  acc.up= acc.up * ord.up (opt= 1)
(case ++,-+)
  acc.lo= acc.up * ord.lo (opt=-1)
  acc.up= acc.up * ord.up (opt= 1)
(case ++,--)
  acc.lo= acc.up * ord.lo (opt=-1)
  acc.up= acc.lo * ord.up (opt= 1)
(caes -+,++)
  acc.lo= acc.lo * ord.up (opt=-1)
  acc.up= acc.up * ord.up (opt= 1)
(case -+,-+)
  acc.lo= min{ acc.lo * ord.up,
         acc.up * ord.lo } (opt=-1)
  acc.up= max{ acc.lo * ord.lo,
         acc.up * ord.up } (opt= 1)
(case -+,--)
  acc.lo= acc.up * ord.lo (opt=-1)
  acc.up= acc.lo * ord.lo (opt= 1)
(case --,++)
  acc.lo= acc.lo * ord.up (opt=-1)
  acc.up= acc.lo * ord.lo (opt= 1)
(case --,-+)
  acc.lo= acc.lo * ord.up (opt=-1)
  acc.up= acc.lo * ord.lo (opt= 1)
(case --,--)
  acc.lo= acc.up * ord.up (opt=-1)
  acc.up= acc.lo * ord.lo (opt= 1)
```

表 1

prc	addrp	subrp	mulrp	divrp	addri	subri	mulri	divri
1	1724	1538	263	61	757	689	126	32
2	1587	1388	115	40	699	625	56	19
3	1428	1282	66	27	636	571	32	13
4	1333	1176	42	20	595	520	21	10
5	1250	1111	30	15	548	487	15	7
6	1136	1020	22	12	512	454	11	6
7	1086	952	17	10	480	429	8	4
8	1030	909	14	8	448	400	7	3

(times/sec)

表 2

参考文献

[1] G.Alefeld and J.Herzberger: Introduction to Interval Computations, Academic Press 1983.

[2] R.T.Gregory and D.L.Karney: A Collection of Matrices for Testing Computational Algorithms, Wiley-Interscience 1969.

[3] R.T.Gregory and E.V.Krishnamurthy, Methods and Applications of Error-Free Computation, Springer-Verlag 1984.

[4] U.W.Kulisch and W.L.Miranker: The arithmetic of the degital computer : a new approach, SIAM Review 28(1986), 1-40.

[5] R.E.Moore: Methods and Applications of Interval Analysis, SIAM publ. 1979.

[6] 王徳人・張連生・㚬乃揚：非線性方程的区間算法，上海科技出版社 1987.

[7] 山本哲朗：数値計算における基本手法“区間演算”について，情報処理学会研究報告 86-NA-17(1986),1-6.

[8] T.Yamamoto: A posteriori componentwise error estimate for a computed solution of a system of linear equations, Publ. Res. Inst. Math. Sci. Kyoto Univ. 18(1982), 521-537.