数理解析研究所講究録
第831巻 1993年 21-32

数式処理と精度保証付き計算の結合

野田松太郎、近藤祐史†
(愛媛大学工学部)
(Matu-Tarow Noda, Sukehumi Kondo)

1. はじめに

数式処理と数値計算を組み合わせる計算法の必要性は古くからいわれていたが、近年数値・数式融合計算（ハイブリッド計算）の提唱 [9, 7] などにより、さらに強い関心をもたれ始めている。ハイブリッド計算は数式処理の環境下での数式処理と数値計算の結合といえる。いきなり数値計算を行うのではなく、前処理部分はもちろんのこと、計算過程の随所に数式処理と数値計算のフィードバックが相互に行われるので、当然高精度の解を得ることが出来る。特に、数値計算のみでは安定した解を見いだせなかった悪条件問題などに対しても十分な精度の結果を得ることが示されている。この面でハイブリッド計算の考え方は極めて高い精度の精度保証付き計算であるといえる。しかし、ハイブリッド計算で全ての悪条件問題を安全に取り扱えるわけではない。数式処理に併用する数値計算に対する誤差の不安を考える必要がある。当然、ここで用いる数値計算を精度保証付き計算に置き換えることができれば、悪条件問題に対しても計算誤差を意識する必要のない結果を得ることが出来るだろう。すなわち、数式処理と数値計算を単に結合するだけでなく、数式処理と区間演算の結合をも考える。区関数を係数とする多項式や有理式に関する数式処理が可能になる。このような考え方を実現する試みは従来からあまり研究されず、小型ハイブリッド処理システム SYNC[6] 上での、PASCAL-SC との結合の例がある程度である。このシステムでは簡単な疑似言語を設定し、両者の結合を試みている。しかし、これは基本としてはファイル経由での結合であり、疑似言語の持つ制約によって限られた範囲の問題にしか利用出来ない。本論では、このような制約をなくし、数式処理と区間演算の結合がはかれるような新しい科学計算システムの構築のための一つの方法を提示し、実際に計算例を示してこのようなシステムの有効性を見ようとする。

区間演算と数式処理を結合することの長所は単に演算途中での数値誤差を保証した結果を得られるだけではない。よく知られているように、区間演算では常に区間の上限、下限の計算を行って結果の区間数を求めなければならない。点を計算する通常の浮動小数による実数演算より、当然はるかに低速である。また、区間演算の欠点の一つとして、数式中で区間数の出現回数が多くなると、結果の区間が広がる [1, 5]。例えば、

$$f = \frac{[2,3] \times [0,1]}{1 - [2,3]}$$

†現在の勤務先:詫間電波高専電子制御工学科

を考える。ここで、$[a_1, a_2]$ は下限 a_1、上限 a_2 の区間数を表わす。単純に計算すると、$f = [-3, 0]$ を得る。しかし、$x = [2, 3]$ のように変数を導入し、上式で 2 度出現する $[2, 3]$ を 1 度しか出現しない次の等価な式に変形できる。

$$f = \frac{[0,1]}{\frac{1}{x} - 1} = \frac{[0,1]}{\frac{1}{[2,3]} - 1}$$

この変形した式から求めると、$f = [-2, 0]$ となり、区間幅の小さな結果を得る。このような操作は単純な数値計算（PASCAL-SC のような数値のみの区間演算パッケージも含む）では不可能で、数式処理を用い分子分母の共通項を求める計算他が必須となる。

開発する科学計算システムは数式処理システムを基本にしている。システムの効率をあげるためには、数式処理システムの細部を知り、区間数を効果的に組み込むことが必要となる。そのため、用いる数式処理システムはソースコードの範囲までの情報を得られるものでなければならない。我々も国産数式処理システムの開発の重要性を以前から指摘してきたが、本論では、開発言語が C 言語でオープン性を追求している risa/asir [8] を取り上げた。このシステムのデータ構造に区間数データ型を付加し、

- 区間数の定義
- 区間数と実数、複素数の間の相互変換と演算
- 区間数間の間の演算
- 区間数を係数に持つ多項式、有理式の演算

などを可能にした。なお、作成システムは SUN ワークステーション上で稼働し、実数（または複素数）と数式処理の結合による計算速度に比して、十分満足いく速度で精度保証付きの計算結果を得ることが出来る。なお、本論は [3, 4] などで述べた内容を整理しなおしたものであることを付記する。

2. 区間数と区間演算の概要

開発したシステムの基本となる区間演算の概要を簡単に述べる。区間演算を構成する区間数には実区間数、複素区間数の別がある。

2.1. 実区間数とその演算

実数全体を $\boldsymbol{R}$ であらわし、

$$A = \{x | a_1 \leqq x \leqq a_2\} \qquad x, a_1, a_2 \in \boldsymbol{R}$$

なる A を実区間数と呼び再び、$A=[a_1,a_2]$ と英大文字で表す（英小文字は実数）。ただし、$a_1 \leqq a_2$ とする。2 つの実区間数 $A=[a_1,a_2]$、$B=[b_1,b_2]$ $(a_1 \leqq a_2, b_1 \leqq b_2)$ の間の演算を次のように定義する。

$$\begin{cases} A+B &= [a_1,a_2]+[b_1,b_2]=[a_1+b_1,a_2+b_2], \\ A-B &= [a_1,a_2]-[b_1,b_2]=[a_1-b_1,a_2-b_2], \\ A\times B &= [a_1,a_2]\times[b_1,b_2]=[\min(a_1b_1,a_1b_2,a_2b_1,a_2b_2),\max(a_1b_1,a_1b_2,a_2b_1,a_2b_2)] \\ A/B &= [a_1,a_2]/[b_1,b_2]=[a_1,a_2]\times[1/b_2,1/b_1] \\ &= [\min(a_1/b_2,a_1/b_1,a_2/b_2,a_2/b_1),\max(a_1/b_2,a_1/b_1,a_2/b_2,a_2/b_1)], \\ &\quad (\text{ただし、}\ 0\notin B). \end{cases}$$

実際に計算機で計算する場合には、結果の実区間数の下限は切捨て、上限は切り上げる。実区間数の加法と乗法はともに結合法則と交換法則を満足する。しかし、分配法則は必ずしも成立しないことはよく知られている。

2.2. 複素区間数とその演算

区間の上、下限が複素数の場合を複素区間数といい、やはり英大文字で区間数を表わす。この定義には矩形型と円盤型の 2 種類がある。

1. 矩形型： (実区間数 $+\ i$ 実区間数)

$$\begin{aligned} A &= A_R+iA_i \\ &= \{x=a_1+ia_2 | a_1\in A_R, a_2\in A_i\} \end{aligned}$$

(ただし、A_R, A_i は実区間数、$i=\sqrt{-1}$ である。)

また、2 つの矩形型複素区間数 A, B の間の演算を次のように定義する。

$$\begin{cases} A+B &= A_R+B_R+i(A_i+B_i), \\ A-B &= A_R-B_R+i(A_i-B_i), \\ A\times B &= A_RB_R-A_iB_i+i(A_RB_i+A_iB_R), \\ A/B &= \dfrac{A_RB_R+A_iB_i+i(A_iB_R-A_RB_i)}{B_R^2+B_i^2}, \\ &\quad (\text{ただし、}0\notin(B_R^2+B_i^2)). \end{cases}$$

2. 円盤型： <複素数、半径>

$$\begin{aligned} A &=< a,r >= \{z\in \boldsymbol{C} |\ |a-z| \leqq r\} \\ &\quad (\text{ただし、}a\in \boldsymbol{C}, r\in \boldsymbol{R}) \end{aligned}$$

2つの円盤型複素区間数を $A =< a, r_a >$, $B =< b, r_b >$ と書くと、それらの演算は次のように定義される。

$$\begin{cases} A+B &= < a+b, r_a+r_b >, \\ A-B &= < a-b, r_a-r_b >, \\ A \times B &= < a \times b, |a|r_b + |b|r_a + r_a r_b >, \\ A \,/\, B &= A \times \dfrac{1}{B}, \\ & \dfrac{1}{B} =< \dfrac{\bar{b}}{b\bar{b}-r_b^2}, \dfrac{r_b}{b\bar{b}-r_b^2} >, \\ & (\text{ただし、}\bar{b}\text{ は }b\text{ の共役複素数}) \end{cases}$$

2種の複素区間数の定義うち、矩形型では複素数計算における実数計算を単純に実区間演算に置き換えた形をしている。虚部の表現にも実区間数を用いるため扱いが容易である。また、2つの区間数の交わりが再び矩形型複素区間数になるという利点がある。一方、円盤型では演算の手間が少なく結合法則が常に成り立つ。しかし、交わりは必ずしも円盤にならず計算に手間がかかる。また円盤型複素区間数の実現のためには実区間数とは別の演算用の関数を用意する必要がある、など計算機での実現には困難がつきまとう。このため、PASCAL-SC などの区間演算用パッケージでも矩形型複素区間数のみを用いている。本システムでもこの点に留意し、矩形型複素区間数のみを考える。

2.3. 区間演算のための諸関数

区間数間の四則演算は上で述べたが、これらを計算機で実現するためには対応する関数をつくる必要がある。四則演算以外にも以下の区間演算関数が必要となる。複素区間数は矩形型を用いるので、$A = A_R + iA_i$ 等となる。ただし、A_R, A_iは実区間数である。

区間数の入力 $A = [a_1, a_2]$

実区間数 ⇒ $\mathrm{itv}(a_1, a_2)$
複素区間数 ⇒ 実区間数の入力を繰り返す。

四則演算

$A * B, * \in \{+, -, \times, /\}$, ただし、/ の場合は $0 \notin B$

区間数の中点 mid (A)

実区間数 ⇒ $(a_1 + a_2)/2$
複素区間数 ⇒ $\mathrm{mid}(A_R) + i\mathrm{mid}(A_i)$

区間数の幅 width (A)

実区間数 ⇒ $a_2 - a_1$
複素区間数 ⇒ $\sqrt{\{\mathrm{width}(A_R)\}^2 + \{\mathrm{width}(A_i)\}^2}$

区間数の絶対値 abs (A)

実区間数 $\Rightarrow$ $\max(|a_1|, |a_2|)$
複素区間数 $\Rightarrow$ $\sqrt{A_R^2 + A_i^2}$

区間数間の距離 distance (A, B)

実区間数 $\Rightarrow$ $\max(|a_1 - b_1|, |a_2 - b_2|)$
複素区間数 $\Rightarrow$ $\mathrm{distance}(A_R, B_R) + \mathrm{distance}(A_i, B_i)$

区間数の下限 inf (A)

実区間数 $\Rightarrow$ a_1
複素区間数 $\Rightarrow$ $\mathrm{inf}(A_R) + i\mathrm{inf}(A_i)$

区間数の上限 sup (A)

実区間数 $\Rightarrow$ a_2
複素区間数 $\Rightarrow$ $\mathrm{sup}(A_R) + i\mathrm{sup}(A_i)$

区間数間の交わり cap (A, B)

実区間数 $\Rightarrow$ $[\max(a_1, b_1),\ \min(a_2, b_2)]$
複素区間数 $\Rightarrow$ $A_R \cap B_R + i(A_i \cap B_i)$

区間数間の結び cup (A, B)

実区間数 $\Rightarrow$ $[\min(a_1, b_1),\ \max(a_2, b_2)]$
複素区間数 $\Rightarrow$ $A_R \cup B_R + i(A_i \cup B_i)$

数の包含

数 $c \in$ 区間数 A $(\text{if } c \in A \text{ then 1 else 0.}) \Rightarrow \mathrm{initv}(A, c)$

有理数からの変換

有理数から区間数への変換時の切り上げ

上の諸関数は引数が実区間数か複素区間数かによって自動的に対応する。数式処理との結合を考えると、これら関数の組み込みが容易でかつ実行が高速である必要がある。

3. 区間演算のためのデータ構造

次に区間数と区間演算を数式処理の環境下に組み込むことを考える。この場合、

1. 区間演算が可能な限り高速に行えること
2. 区間数を係数に持つ多項式や有理式を有理数または実数係数のように扱えること
3. 数式処理結果と区間演算、区間演算の結果の数式処理での再利用が容易であること

4. 区間数や区間演算のための関数の組み込みの容易さのために、基本となる数式処理システムのソースコードの変更個所が少ないこと

などを満たす必要がある。これら（特に 1 ～ 3 ）を満たせば、PASCAL-SC のような区間演算パッケージの使用に比してはるかに区間演算の実際的問題への適用が可能になる。このためには、区間数の定義を数式処理システムの多様なデータ構造のうち、どのレベルにおけばよいかを明確にする必要がある。

通常の数値計算のシステムで扱うデータは整数、実数（単精度、倍精度）、文字型、論理型を基本としたもののみでよい。複素数は 2 つの実数を実部、虚部に割り当てればよい。区間数も同様であり、実数型データをその上限、下限に対応させて定義する。したがって、通常の区間演算パッケージでは区間数を含む式の処理は数値計算のみになる。これに対して、数式処理システムが持つデータ構造は非常に複雑である。伝統的に記号処理分野からの影響が強いこともあり、実数なども多くのデータ型のごく一部にすぎない。有理数演算を行う場合も、分子、分母の整数演算のみで実現するのではなく、別個に有理数のデータ型を持つようになっている。同様に、多項式、有理式、行列、リスト等々がシステムのデータ構造に含まれている。これらを区別するために、多くのビットが使われ（フラグと呼ぶ）、1 語は数値計算に比して長大になり計算の低速化を招く。当然、実数計算といえども FORTRAN の数値計算によるものよりは、高精度ではあるが低速である。したがって、区間数を数式処理システムのデータ構造に組み込むには、十分な配慮を払わない限り計算速度が低下する可能性がある。なお、単純に区間数を 2 つの実数で定義するのみにすると、数値計算、数式処理を問わず、区間演算結果を得るのに複数回の実数の四則演算が必要となり、計算速度は実数計算に比して非常に低速になる。少なくとも、区間の上限、下限の計算を各々実数計算するので、単純な実数演算の少なくとも 2 倍以上の計算時間を必要とするのは明かである。

数式処理システム risa/asir に区間数を組み込むことを考える。考慮すべき点は

- データ構造のどの位置に区間数データ型をおくか
- 区間数データをどのように格納するか

である。本来のシステムのデータ構造の概略は次図 1 のように表される [8]。

図 1 が意味することは、例えば多項式や有理式は有理数以下の数を係数にするが、ベクトルを係数にできない等である。したがって、区間数を数の概念に含めて組み込めば、多項式の係数にも区間数を使用することが可能になる。数のデータ構造中、2 と 3 の間、3 と 4 の間、4 の後ろへ挿入が考えられる。高速さと関数の組み込み時の risa/asir のソースコードの変更を最小にするには 3 と 4 の間に挿入する方法が最良であることがわかった。また区間数データの格納方法は、ビット長が少なく計算の高速化を目指して、

区間の下限、上限に数値計算と同様の倍精度浮動小数を利用する方法

を採用した。これにより実区間数はシステム内で次のように表現される。
risa/asir 複素数のデータ構造は実部、虚部がそれぞれ複素数以外の数へのポインタで構成されている。また、矩形型複素区間数は **2.2.** でも述べたように複素数の実部、虚部をそれぞれ実区間数に置き換えた形をしているため、複素区間数を実現するのに既存の複素数のデータ構造をそのまま利用出来る。このように、既存の数式処理システムのデータ構造を可能な限り踏襲することによって、**2.3.** の区間演算用の関数を実現する場合に、基本システムのソースコードの書換を最小限にとどめる得た。

4. システムの計算例

このようにして開発した新しいシステムで可能な計算、及び計算の効率について簡単に触れる。システムでは、区間数、普通の数、記号などが混在する多項式や行列の数式処理が可能になる。このような特色を生かした新しいタイプのアルゴリズム（区間数や記号を含めた数式の因数分解…）の可能性は今後の課題である。以下では、ここで開発したシステムの効率や応用の例題を述べる。それらの詳細は [3, 4] にある。

4.1. 連立代数方程式の求解

まず、連立代数方程式

$$\begin{cases} f_1(x_1, x_2) &= x_1^2 + x_2^2 - 1 = 0, \\ f_2(x_1, x_2) &= x_1^2 - x_1 x_2 - x_2^2 - 3 = 0 \end{cases} \tag{1}$$

を複素数の Newton 法（①）

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}, \quad k \geq 0,$$

および、区間数に拡張した複素区間 Newton 法（②）

$$X^{(k+1)} = m(X^{(k)}) - \frac{f(m(X^{(k)}))}{F'(X^{(k)})}, \quad k \geq 0,$$

（ただし、$m(X)$ は区間 X の中点を表す。）

で解き、解の精度と演算回数を比較する。①の計算は risa/asir の複素数演算機能を用い、②はここで開発システムで行った。初期値は $x_1^{(0)} = x_2^{(0)} = 1 + i$ またはその区間数化したものを用いた。SUN/ELC 上で求めた結果をそれぞれ表 1 に示す。　区間演算を行なうと精度保証付きの解を求めることが出来る。収束するまでに要する反復回数も複素数計算、複素区間演算で等しい。しかも計算時間は②の区間演算では①の複素数計算に比し、1.3 倍程度である。これは **3.** で述べた予測よりはるかに高速である。これは基本の数式

表 1: 方程式 (2) の解の 1 つの精度と演算回数

方法	解	精度保証	Iter.	計算時間
複素 Newton 法 (risa/asir)	$x_1 = 1.3207676521629061 + 0.21077758656915082i$ $x_2 = -0.31179094839324300 + 0.89286818484026964i$	—	6	150 msec
複素区間 Newton 法	$x_1 = [\underline{1.3207676521629061}, \underline{1.3207676521629065}] + [\underline{0.21077758656915074}, \underline{0.21077758656915096}]i$ $x_2 = [\underline{-0.31179094839324317}, \underline{-0.31179094839324289}] + [\underline{0.89286818484026941}, \underline{0.89286818484026986}]i$	$4.44E-16$	6	200 msec

Iter. は収束するまでの反復回数を表す。

処理システムのデータ構造に着目し、高速性を保つことに十分な配慮を払って区間数データ型を挿入した成果である。なお、同じ計算を SUN/ELC 上の FORTRAN で行うと、

$x_1 = 1.3207676521629064 + 0.21077758656915083i$
$x_2 = -0.31179094839324301 + 0.89286818484026953i$
反復回数 (Iter.): 8 回
計算時間: 11 msec

を得る。**3.**に述べたように通常の数値計算は数式処理システム上での数値計算より高速である。ただし、単純な数値計算では問題によっては精度上の問題点があり、かつ計算結果を得るための入力の困難、入力された式への操作、計算結果の再利用などの面で数式処理を経由するものより劣る。また、計算速度に関して付け加えると、区間演算では特徴的な Krawczyk–Moore 法を用いるた場合も、表 1の場合と同じような数計算、区間数計算間の速度比が求まる。[3]

4.2. 梯子型抵抗回路の解析

奥村らは区間演算を電気回路の中の梯子型抵抗回路の解析に適用している [2]。この問題を数式処理を併用した本システムで解析してみる [4]。数式処理を用いることにより、**1.** で述べた変数の出現の問題に対応し得る可能性を探る。図 3のような 3 段の梯子型抵抗回路を考える。 [2] でも述べられているが、このような回路を実際に製作した場合に抵抗器の値にズレがあり、正確な解析は困難となる。この問題は直流電源と抵抗器で構成さ

れる回路であり実区間数の使用のみでよいが、交流電源でコイルやコンデンサなどを含む電気回路の解析では複素区間数計算が必要である。図 3 のような回路の回路方程式はキルヒホッフの法則により次の連立 1 次方程式になる。

$$\begin{cases} V_1G_1 + (V_1 - V_2)G_4 - J = 0 \\ V_2G_2 + (V_2 - V_1)G_4 + (V_2 - V_3)G_5 = 0 \\ V_3G_3 + (V_3 - V_2)G_5 = 0 \end{cases}$$

ここで、J, G_i, $i = 1, \cdots, 5$ は $J = 10[1-\varepsilon, 1+\varepsilon]$, $G_i = [1-\varepsilon, 1+\varepsilon]$, $i = 1, \cdots, 5$, $0 \leqq \varepsilon \leqq 0.1$ とする。この方程式を解く場合次の 2 通りの方法が考えらえる。

1. 数値を代入して、ガウス消去により数値解を求める。

2. 記号的にガウス消去をして、数値を代入する。

奥村らは前者を単純な数値計算で行うと満足できる解を求められないので、区間演算で解を得ている。さらに進めて、後者の数値代入を区間数の代入に置き換えた計算が本システムでは可能になる。まず、方程式の記号解を数式処理により求めると、

$$\begin{cases} V_1 = \dfrac{(JG_3 + JG_5)G_4 + (JG_3 + JG_5)G_2 + JG_3G_5}{((G_3 + G_5)G_4 + (G_3 + G_5)G_2 + G_3G_5)G_1 + ((G_3 + G_5)G_2 + G_3G_5)G_4} \\ V_2 = \dfrac{(JG_3 + JG_5)G_4}{((G_3 + G_5)G_4 + (G_3 + G_5)G_2 + G_3G_5)G_1 + ((G_3 + G_5)G_2 + G_3G_5)G_4} \\ V_3 = \dfrac{JG_4G_5}{((G_3 + G_5)G_4 + (G_3 + G_5)G_2 + G_3G_5)G_1 + ((G_3 + G_5)G_2 + G_3G_5)G_4} \end{cases} \tag{2}$$

となる。ここで、この記号解に区間数の数値を代入するが、変数の出現回数の問題がある。記号解 (2) 中の各変数の出現回数を最小にするため数式処理を活用する。結果は、

$$\begin{cases} V_1 = \dfrac{J}{G_1 + \dfrac{1}{\dfrac{1}{G_4} + \dfrac{1}{G_2 + \dfrac{1}{\dfrac{1}{G_5} + \dfrac{1}{G_3}}}}} \\ V_2 = \dfrac{J}{G_1 + (G_2 + \dfrac{1}{\dfrac{1}{G_5} + \dfrac{1}{G_3}})(1 + \dfrac{G_1}{G_4})} \\ V_3 = \dfrac{J}{(1 + \dfrac{G_3}{G_5})(G_2(1 + \dfrac{G_1}{G_4}) + G_1) + G_3(1 + \dfrac{G_1}{G_4})} \end{cases} \tag{3}$$

となる。$\varepsilon = 0 \sim 0.1$ の範囲で次の 3 種の結果を比較する。

1. 区間数の数値を代入して、ガウス消去により数値解を求める場合、

2. 記号解 (2) に区間数の数値を代入する場合、

3. 各変数の出現回数を最小にした記号解 (3) に区間数の数値を代入する場合。

結果の概略を $\varepsilon = 0.05, 0.1$ で得られた区間幅に対し次表に簡単にまとめる。

解法	V_1		V_2		V_3	
	ε		ε		ε	
	0.05	0.1	0.05	0.1	0.05	0.1
1	1.216	2.524	1.26	2.88	1.82	4.225
2	1.658	4.56	1.215	2.565	1.3	2.73
3	0.76	1.442	0.54	1.125	0.78	1.485

V_1, V_2, V_3 のいづれの場合も 3に区間数値を代入したもの（**3** の解法）が最も狭い範囲を示しており、要求結果に近い。このような操作が可能であることは数式処理と区間演算を結合したシステムを利用する長所であると言える。なお、ここで変数の出現を少なくするために用いた手法はきわめて原始的であるが、より規模の大きな問題に対しては構成的アルゴリズムを設計する必要がある。

5. まとめ

本論では数式処理システムへの区間演算を付加した科学計算システムの開発を行なった。このシステムでは、数値計算と数式処理を結合したハイブリッド計算の手法にさらに誤差保証付き計算を加えた結果を求めることが出来る。より具体的には、数式処理システムのデータ構造に区間数を導入することにより、誤差の不安がない計算を比較的高速に行う可能性を示した。本システムの特徴には、

1. 浮動小数計算とは異なり、誤差保証付き計算を行うことが出来る。

2. 計算結果を再び数式処理のアルゴリズムで簡単化などを行うことが出来る。

3. 係数が区間数の多項式や有理式あるいは行列などの数式処理が可能である。

4. 計算時間は数式処理システム上での数値計算に比較して 1.3 倍程度と高速である。

などがある。まだ十分には計算例を示していないが、今後本システムを悪条件問題などに適用することによって、今まで困難だった多くの問題を精度保証付きで数値的に、あるいは数式的に求め得る可能性がある。

なお、システムの完備のため今後早急に解決すべき点は三角関数、指数関数、対数関数などの関数への区間演算の対応、変数の出現回数を減らすような問題に適合する構成的アルゴリズムの開発など山積している。しかし、ここに開発したシステムは **USSC**(Ultimate System for Scientific Computation) への出発点を与えたといえるであろう。

参考文献

[1] G. Alefeld and J. Herzberger, "Introduction to Interval Computations", (Translated by J. Rokne), Academic Press, New York, 1983.

[2] K. Okumura and A. Kishima, On Applications of Interval Arithnetic to Circuit Analysis, 数理解析研究所講究録 673 「自己検証的算法とその応用」, 1988.

[3] 近藤祐史、野田松太郎, 数式処理と区間演算との結合, 日本数式処理学会第 1 回研究報告会報告集,1993(印刷中).

[4] 近藤祐史、野田松太郎、数式処理と区間演算との結合–複素区間数の場合–, 数理解析研講究録「数式処理の数学研究への応用」, 1993(出版予定).

[5] R.E. Moore, "Methods and Applications of Interval Analysis", SIAM Studies in Applied Mathematics, SIAM, Philaderphia, 1979.

[6] 野田松太郎、岩下英俊, パソコンで稼働する小型ハイブリッドシステム SYNC の設計, 情報処理学会論文誌, Vol.30, No.4, pp. 419–426, 1989.

[7] M.–T. Noda and T. Sasaki, Approximate GCD and its application to ill–conditioned algebraic equations, Journal CAM, Vol.38, pp.335–351, December, 1991.

[8] M. Noro and T. Takeshima, Risa/Asir : A Computer Algebra System, in Proc. ISSAC'92, July 26–29, 1992 at Berkeley, CA.

[9] T. Sasaki and M.–T. Noda, Approximate Square–free Decomposition and Root–finding of Ill–conditioned Algebraic Equation, J. Inf. Proces, Vol.12, pp.159–168, 1989.

- 0. 零 (0)
- 1. 数 ・・・
  - 0. 有理数 ・・・
    - 整数
    - 分数
  - 1. 倍精度浮動小数
  - 2. 代数的数
  - 3. 多倍長小数
  - 4. 複素数
- 2. 多項式
- 3 ～ 8　有理式、リスト、ベクトル、行列、文字列、構造体

図 1: risa/asir のデータ構造

フラグ	下限: double 型のデータ	上限: double 型のデータ

160 bits

図 2: 実区間数の格納方法

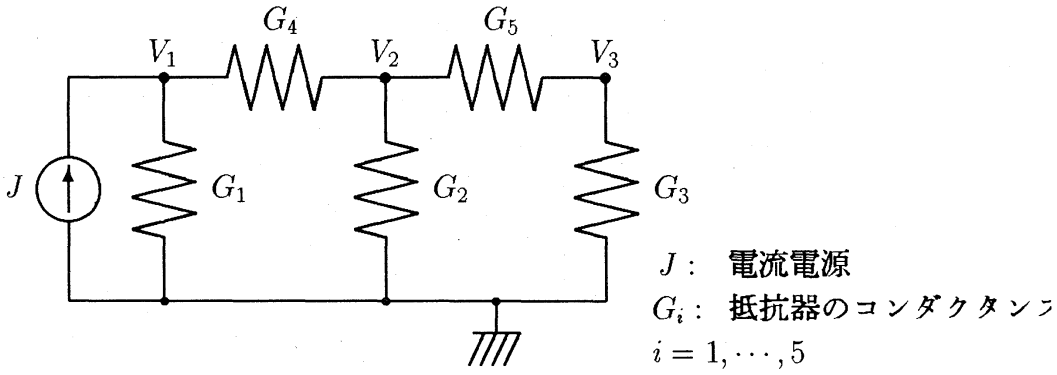

図 3: 梯子型抵抗回路