アルゴリズム 48－13
（1995．11．17）

非凸多角形区間演算とそのインプリメント

太田 有三　田川 聖治　羽根田 博正
神戸大学 工学部 電気電子工学科

本文では，非凸多角形区間演算 (Non-convex Polygon Interval Arithmetic: NPIA) の定義するとともにそのインプリメントの概略について述べる．NPIA は，複素平面の必ずしも凸でない多角形の集合の上で多角形に対する和，積，逆集合を定義しており，その演算結果は対応する演算の値集合を含み，そのε近傍に含まれるような多角形を与えるものである．ここでは，計算幾何学的方法に基づく NPIA のインプリメント法を示す．この方法における中心課題は，多数の多角形が与えられた時，それらの和集合の外部境界を求める問題である．ここでは，線分の交差判定法を利用した方法を用いている．NPIA は，不確かさを持つパラメータ qを含む制御系の特性多項式または伝達関数 $f(s,q)$ を考えるとき，その値集合の推定を得るのに用いられる．これらの推定を得ることは，ロバスト制御系の解析・設計に有用である．

On the Implementation of Non-convex Polygon Interval Arithmetic

Yuzo Ohta　Kiyoharu Tagawa　Hiromasa Haneda
Department of Electrical and Electronics Engineering, Kobe University

In this paper, a method to implement Non-convex Polygon Interval Arithmetic (NPIA) is presented. NPIA is an arithmetic defined on the set of all (non-convex) polygons in the complex plane. The operations (that is, addition, multiplication and inversion) gives a polygon contained an ε neighborhood of the value set of the corresponding operation. The main issue of this approach is to compute the outer boundary of the union of many polygons. An implementation method based on the computer geometry the geometric intersection algorithm is presented. NPIA can give an estimate region which is included an ε neighborhood of the value set of a given function $f(s, q)$ which includes uncertain parameters q.

1　はじめに

本文では，複素平面の必ずしも凸でない多角形の集合の上で和，積，逆集合が定義される多角形区間演算 (Non-convex Polygon Interval Arithmetic: NPIA[1]) のインプリメントの概略と実際にインプリメントした場合に丸め誤差によって生ずる問題点について述べる．NPIA は，不確かさを持つパラメータを含む制御系の特性多項式または伝達関数 $f(s,q)$ を考えるとき，その値集合

$$V(f;\ s,\ Q) = \{f(s,\ q) \mid q \in Q\}$$

の推定を得るのに用いられる．ここで，Q は，不確かさを含むパラメータ qの不確かさの領域を表す集合である．従来の多角形区間演算 (PIA) は，複素平面上の凸多角形の集合の上で定義されてきたが [2] [3]，PIA を用いた値集合 $\mathbf{V}(f;\ s,\ Q)$ の推定 $f_{PIA}(s,\ Q)$ は，値集合自体と一致するとは限らないので，より精確な推定が必要である場合には，パラメータ空間 Q の分割を繰り返す必要がある．fがqの多重線形関数の場合の Q の分割方法は，文献 [5] で提案されているが，対象とする $f(s,q)$ によっては，値集合またはそのε近傍が 0 を

含むか否かの判定に非常に多数の分割を必要とする場合がある．この原因は，PIAでは，演算結果が凸多角形になるようにしていたことにある．そこで，演算結果が必ずしも凸でない多角形になることを許し，演算結果が値集合のε近傍に含まれるような多角形であるようにするために，従来のPIAを拡張したものが，必ずしも凸でない多角形の集合上で定義されるNPIA[1]である．

このNPIAのインプリメントにおける中心的な問題は，多数の多角形の和集合を求める問題になるが，本文で考える問題では，この多角形が共通の辺を有している特徴があり，この特徴を活かした方法の概略と実際にインプリメントした場合に丸め誤差によって生ずる問題点について述べる．

2 NPIAの定義

多角形区間演算(NPIA[1])は，伝統的な区間演算を複素平面$\mathbf{C}$上の凸でない多角形も含めた全ての多角形から成るクラス$\mathcal{P}$に対して適用できるように一般化したものである．このクラス$\mathcal{P}$に対する和，積，逆集合を表す演算子を，それぞれ，$\oplus$，$\odot$，$(\cdot)^{\ominus}$で表す．$\mathcal{P}$の要素に対してこれらの演算を施したものが再び$\mathcal{P}$の要素となるように代数系NPIA$(\mathcal{P};\oplus,\odot,(\cdot)^{\ominus})$は定義される．ただし，ここでは，除算$w/z$は，$w\cdot z^{-1}$のように，逆数を乗ずることと考えている．

パラメータ$q\in Q$を含む関数$f(q)$が与えられたとき，その値集合は，

$$\mathbf{V}(f:\ Q)=\{f(q)|\ q\in Q\}$$

で定義される．以下では，特に，$q=[z\ w]^T$，$Q=Z\times W$，$f(q)=f(z,w)=z+w$，$f(z,w)=z\cdot w$の場合や，$Q=Z$，$f(q)=f(z)=z^{-1}$である場合を考える．一般に，$Z,W\in\mathcal{P}$が単純な領域な場合でも，$\mathbf{V}(z\cdot w;\ Z,W)$は，単純な領域とは限らないので，NPIAの演算結果が値集合の外部境界で囲まれた領域を含み，そのε近傍に含まれるような多角形になるようにNPIAを定義する．値集合の外部境界で囲まれた領域自体を計算するのではなく，そのε近傍に含まれる多角形を求めるのは，後に示すように，積や逆集合の境界が曲線になりうるためで，計算の効率の改善と記憶容量の低減のためである．便宜上，集合$V\subseteq\mathbf{C}$に対して，そのε近傍およびVの外部の境界で囲まれる領域を，それぞれ，$\mathcal{N}(V,\varepsilon)$，outbd$[V]$で表すとき，次のような性質を持つ演算を定義する．

A) $\mathrm{outbd}[\mathbf{V}(z+w;\ Z,W)]=Z\oplus W$ (1)

B) $\mathbf{V}(z\cdot w;\ Z,W)\subseteq Z\odot W$
$\subseteq\mathcal{N}(\mathrm{outbd}[\mathbf{V}(z\cdot w;\ Z,W)];\ \varepsilon)$ (2)

C) $\mathrm{outbd}[\mathbf{V}(z^{-1};\ Z)]\subseteq Z^{\ominus}$
$\subseteq\mathcal{N}(\mathrm{outbd}[\mathbf{V}(z^{-1};\ Z)],\varepsilon)$ (3)

以下では，特に断らない限り，$Z,W\in\mathcal{P}$を，それぞれ，n角形，m角形とし，逆集合$Z^{\ominus}$を考える場合は，Zは，0を含まないものとする．また，Zの端点$\{z_1,z_2,\ldots,z_n\}$は，反時計回りに並べられているものとする．Wの端点についても同様である．Z,Wの境界を，それぞれ，$\partial Z,\partial W$で表し，Z,Wの辺の集合を，それぞれ，$E_Z=\{L_Z^i|\ i\in[1..n]\}$，$E_W=\{L_W^j|\ j\in[1..m]\}$で表す．明らかに，$\partial Z=E_Z$，$\partial W=E_W$である．

ところで，$z\in\mathbf{C}$と$W\in\mathcal{P}$を与える時，$z\oplus W$と$z\odot W$を，それぞれ，$\{z+w_1,z+w_2,\ldots,z+w_m\}$と$\{z\cdot w_1,z\cdot w_2,\ldots,z\cdot w_m\}$を反時計回りに並べた多角形とすれば，

$$z\oplus W=\mathbf{V}(z+w;\ \{z\},W),\tag{4}$$

$$z\odot W=\mathbf{V}(z\cdot w;\ \{z\},W)\tag{5}$$

が成り立つ事に注意しておこう．

従来の(凸多角形の上での)PIAにおける和と積は以下のように定義されている．

定義 1 凸多角形の集合を$\mathbf{P}_{\mathrm{conv}}$で示す．端点の集合が$\mathrm{ext}(Z)=\{z_1,z_2,\ldots,z_m\}$，$\mathrm{ext}(W)=\{w_1,w_2,\ldots,w_n\}$で与えられるような2つの$\mathbf{P}_{\mathrm{conv}}$の要素$Z$と$W$に対して，$Z$と$W$の和と積を

$$Z+W=\mathrm{conv}[\{z_i+w_j|\ z_i\in\mathrm{ext}(Z),\ w_j\in\mathrm{ext}(W)\}]\tag{6}$$

$$Z\bullet W=\mathrm{conv}[\{z_i\cdot w_j|\ z_i\in\mathrm{ext}(Z),\ w_j\in\mathrm{ext}(W)\}]\tag{7}$$

で定義する．■

$\circ=+$または，$\cdot$であるとすると，一般に，

$$\partial\mathbf{V}(z\circ w;\ z\in Z,w\in W)$$

$$\subseteq\{z\circ w|\ z\in\partial Z,\ w\in\partial W\}\tag{8}$$

が成り立つ[6]．ここで，$\partial\mathbf{V}$は，$\mathbf{V}$の境界を示す．明らかに，

$$\partial\mathbf{V}(z\circ w;\ Z,W)$$

$$\subseteq\{z\circ w|\ z\in L_Z^i\in E_Z,\ w\in L_W^j\in E_W\}\tag{9}$$

が成り立つ．従って，上式の右辺の各要素(集合)を求め，それらの和集合の外部境界を求めればよい．

まず，和の演算について考える．和については，

$$\mathbf{V}(z+w;\ L_Z^i,L_W^j)=L_Z^i+L_W^j\tag{10}$$

が成り立つから [2], $\mathbf{V}(z+w;\ z\in Z,\ w\in W)$ の外部境界は，線分である．従って，

$$Z \oplus W = \mathrm{outbd}[(\bigcup_{z\in \mathrm{ext}(Z)} z \oplus W) \bigcup (\bigcup_{w\in \mathrm{ext}(W)} Z \oplus w)] \quad (11)$$

と定義される [1]．なお，$Z, W \in \mathbf{P}_{\mathrm{conv}}$ の場合は，$\mathbf{V}(z+w; Z, W)$ は凸多角形であり，$\mathbf{V}(z+w; Z, W) = Z \oplus W = Z + W$ であるが，そうでない場合には，$\mathbf{V}(z+w; Z, W)$ は単純でない多角形になる場合があり，一般に，$\mathbf{V}(z+w; Z, W) \subseteq Z \oplus W = \mathrm{outbd}[\mathbf{V}(z+w; Z, W)]$ である．

積の演算について定義するには，次の補題 [6] が必要である．

補題 1 Z, W の任意の辺 L_Z^i, L_W^j を考える．ここで，$L_Z^i = \mathrm{conv}[z_p, z_s]$, $L_W^j = \mathrm{conv}[w_q, w_r]$ である．

$$v(\alpha,\beta) = [z_p + \alpha(z_s - z_p)] \cdot [w_q + \beta(w_r - w_q)] \quad (12)$$

$$J(\alpha,\beta) = \begin{vmatrix} \frac{\partial \mathrm{Re}[v(\alpha,\beta)]}{\partial \alpha} & \frac{\partial \mathrm{Re}[v(\alpha,\beta)]}{\partial \beta} \\ \frac{\partial \mathrm{Im}[v(\alpha,\beta)]}{\partial \alpha} & \frac{\partial \mathrm{Im}[v(\alpha,\beta)]}{\partial \beta} \end{vmatrix} \quad (13)$$

$$C(L_Z^i, L_W^j) = \{v(\alpha,\beta) |\ \alpha,\beta \in [0,1],\ J(\alpha,\beta) = 0\} \quad (14)$$

とおくと，

$$\partial \mathbf{V}(z \cdot w; L_Z^i, L_W^j) \subseteq L_Z^i \circledcirc w_q \bigcup L_Z^i \circledcirc w_r \bigcup z_p \circledcirc L_W^j \bigcup z_s \circledcirc L_W^j \bigcup C(L_Z^i, L_W^j) \quad (15)$$

が成り立つ．■

簡単のため，$L_Z^i = \mathrm{conv}[z_0, z_1]$, $L_W^j = \mathrm{conv}[w_0, w_1]$ とおくとき，$U = L_Z^i \bullet L_W^j$ の端点 $\{u_{ij} = z_i \cdot w_j\}$ が，左下の点を中心に反時計周りにソートされているとする．このとき，$u_{00} - u_{01} - u_{11} - u_{10}$ のように，添字 $\{ij\}$ の差が1づつになっている場合は，$J(\alpha,\beta) = 0$ とはならないので，$C(L_Z^i, L_W^j)$ は，存在しない．一方，$u_{00} - u_{11} - u_{10} - u_{01}$ のように添字 $\{ij\}$ の差が2になる場合があるときや U の端点数が3以下である場合には，$J(\alpha,\beta) = 0$ となる $\alpha, \beta \in [0,1]$ が存在する．なお，$J(\alpha,\beta)$ は，α と β の1次関数となり，α - β 平面における直線が得られる．この直線が正方形 $(0,0)-(0,1)-(1,1)-(1,0)$ をどのように横切るかということによって，場合分けが必要であるが，例えば，$L_Z = \mathrm{conv}[-1+j2, 1-j]$, $L_W = [-2+j, 1+j2]$ であるとき，図1のようになる．

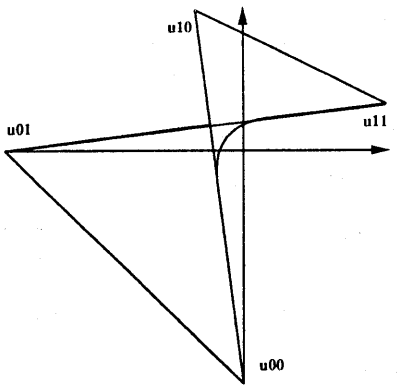

図 1: $L_Z Z \circledcirc L_W$

このように，$C(L_Z^i, L_W^j)$ は，凹な部分となるので．$C(L_Z^i, L_W^j)$ の両端の点と複数の適当な中間の点を選んでできる折れ線を $C_L(L_Z^i, L_W^j)$ で表し，

$$L_Z^i \circledcirc L_W^j = \mathrm{outbd}[L_Z^i \circledcirc w_q \bigcup L_Z^i \circledcirc w_r \bigcup z_p \circledcirc L_W^j \bigcup z_s \circledcirc L_W^j \bigcup C_L(L_Z^i, L_W^j)] \quad (16)$$

とおくと，これは多角形であり，

$$Z \circledcirc W = \mathrm{outbd}[\bigcup_{L_Z^i \in E_Z,\ L_W^j \in E_W} L_Z^i \circledcirc L_W^j] \quad (17)$$

と定義される．

逆集合を定義するためには，次の補題 [2] が必要である．

補題 2 $Z \in \mathcal{P}$ とし，Z は，0 を含まないとする．$\mathrm{ext}(Z) = \{z_1, z_2, \ldots, z_n\}$ とし，Z の辺を，$L^1, L^2, \ldots, L^n$ とする．ここで，$L^p = \mathrm{conv}[z_p, z_s]$ であり，$p < n$ のとき，$s = p+1$ で，$p = n$ のとき，$s = 0$ とする．$w_p = z_p^{-1}$, $p = 1, 2, \ldots\ n$, $\Gamma^p = \{w = z^{-1} |\ z \in L^p\}$ とすると，次のことが成り立つ．

A) もし，$z_p, z_s, 0$ が1直線上にあれば，$\Gamma^p = \mathrm{conv}[w_p, w_s]$ である．

B) もし，$z_p, z_s, 0$ が1直線上になければ，Γ^p は，$w_p, w_s, 0$ を通る円の一部（円弧）で，Γ^p の端点は，w_p と w_s であり，0 を含まない．

C) $\mathbf{V}(z^{-1}; Z) = \{w = z^{-1} |\ z \in Z\}$ は，円弧 $\Gamma^1, \Gamma^2, \ldots, \Gamma^n$ で囲まれた領域である．■

式 (8) と同様に，$V = \mathbf{V}(z^{-1};\ z \in Z)$ の場合も，

$$\partial V \subseteq \{z^{-1} |\ z \in \partial Z\} \quad (18)$$

が成り立つ．

また，$\mathbf{V}(z^{-1};\ L_Z^i)$ を隣接する幾つかの3角形で囲むようにすることにより，与えられた$\varepsilon > 0$に対して，

$$\mathbf{V}(z^{-1};\ L_Z^i) \subseteq \mathcal{N}_{Pinv}(L_Z^i,\ \varepsilon)$$

$$\subseteq \mathcal{N}(\mathbf{V}(z^{-1};\ L_Z^i);\varepsilon) \tag{19}$$

を満たす多角形$\mathcal{N}_{Pinv}(L_Z^i,\ \varepsilon)$を作ることができる．従って，逆集合については，

$$Z^{\ominus} = \mathrm{outbd}[\{\mathcal{N}_{Pinv}(L_Z^i);\varepsilon)|\ L_Z^i \in E_Z\}] \tag{20}$$

と定義する．

$f(q_1,\cdots,q_\ell)$をℓ個の変数の有理関数とする．与えられた関数に対して，和，積，逆数を求める演算を行う順序をある順序に定めたものをその関数の表現と呼ぶ．表現を考えるのは，同じ関数であっても演算の順序を変えることにより，得られる結果が異なるものになる可能性があるからである．表現f_nにおいて，各変数q_iが一度しか現れない場合，f_nは，fの完全分解可能なTSDに対応する表現と呼ぶ．表現$f_n(q_1,\cdots,q_\ell)$において，変数q_iを多角形Q_iに置き換え，通常の和，積，および逆数を求めるの演算を$\mathbf{P}_{\mathrm{simpl}}$における和，積，および逆集合の演算に置き換えたものを多角形の関数と定義し，それを，$\mathbf{P}(f_n;\ Q_1,\cdots,Q_\ell)$で表す．

$$\mathbf{V}(f:\ Q_1,\cdots,Q_\ell) \subseteq \mathbf{P}(f_n:\ Q_1,\cdots,Q_\ell)$$

が成り立つことは，容易に示せるが，一般には，

$$\mathbf{P}(f_n:\ Q_1,\cdots,Q_\ell)$$

$$\subseteq \mathcal{N}(\mathbf{V}(f:\ Q_1,\cdots,Q_\ell);\ \varepsilon) \tag{21}$$

は成立するとは限らないが，f_nが完全分解可能なTSDに対応する表現である時には，上式が成り立つ．

3 NPIAのインプリメント

式(11), (17), (20)からわかるように，$Z \oplus W$, $Z \ominus W$, $Z^{\ominus}$を計算するためには，幾つかの多角形の集合$\{Z_k\}$が与えられたとき，$\mathrm{outbd}[\{Z_k\}]$を計算することが必要となる．

このための1つの方法として，2つの多角形の和集合を求める操作を繰り返すことが考えられるが，我々の応用としている不確かさを含むパラメータを持つ特性多項式や伝達関数の値集合のε近傍に含まれる多角形を得る問題では，2つの多角形の和集合を求めて得られる多角形の辺の大部分は最終的な多角形の辺ではないので，効率が良くない．また，最初から全ての多角形が与えられていることを利用した方が得策であるということもあり，次のようなm個の線分の交差判定法を利用した方法を採用している．

辺は，全部でm本あり，k番目の辺を$L^k = \mathrm{conv}[z_p^k, z_s^k]$とする．また，$x_p^k = \mathrm{Re}[z_p^k]$, $y_p^k = \mathrm{Im}[z_p^k]$, $x_s^k = \mathrm{Re}[z_s^k]$, $y_s^k = \mathrm{Im}[z_s^k]$とする．今，$\{x_p^k\}$を小さい順にソートしたとして，その結果，等しくないx_p^kの数がr個で，$\{x_1 = x_{p_1}^{k_1},\ x_2 = x_{p_2}^{k_2},\ \ldots,\ x_r = x_{p_r}^{k_r}\}$となったとする．

説明を簡単にするために，次のようなデータ構造とアルゴリズムを考える．

```
edge = record
        x_ℓ, y_ℓ, x_r, y_r: real
    end;
```

という型を考える．ここで，edgeは，辺L^kを記憶するためのデータ構造であり，$(\mathrm{x}_\ell,\ \mathrm{y}_\ell)$, $(\mathrm{x}_r,\ \mathrm{y}_r)$は，$\mathrm{x}_\ell \leq \mathrm{x}_r$となるように辺$L^k$の端点$z_p^k$, z_s^kと対応させている．但し，垂直な辺の場合は，$\mathrm{y}_\ell < \mathrm{y}_r$となるように対応させる．

大きさが，mのedgeを要素とする1次元配列edge_array，大きさがrの1次元実数配列x_array，列数rで行数mの2次元実数配列slab_arrayを用意する．edge[k]は，辺L^kに対応し，x_array[i] $= x_i$である．辺L^kが，$x_i \in [\mathrm{x}_\ell,\ \mathrm{x}_r]$を満たす時，直線$x = x_i$と辺$L^k$の交点$(x_i,\ y_i)$の虚数部$y_i$をslab_array[$i$][$k$]に記憶する．$L^k$が垂直な辺である場合には，slab_array[$i$][$k$]には，$\mathrm{y}_\ell$の値を記憶する．また，$x_i \notin [\mathrm{x}_\ell,\ \mathrm{x}_r]$の時には，slab_array[$i$][$k$] $= \infty$とする．

slab_array[1][k]の値が最小である添字をk_1とするとき，$\mathrm{Re}[z_1] = \mathrm{x\_array}[1]$, $\mathrm{Im}[z_1] = \mathrm{slab_array}[1][k_0]$となる点を$z_1$とする．この$z_1$は求める多角形の左下隅の点である．

アルゴリズムは，次のようになる．

Algorithm get_outer_boundary

1. $i := 1$, $z := z_1$とする．
2. zを，Zの端点として登録する．
3. slab_array[i][k] $= \mathrm{Re}[z]$となる垂直でない辺とslab_array[i][k] $\leq \mathrm{Re}[z] \leq$ edge_array[k].y_rを満たす垂直な辺の中から最も右側にある辺L^kを求める．
4. もしL^kが，右方向に進む辺ならば，5.へ，そうでないならば，8.へ．
5. もし直線$x = \mathrm{x_array}[i]$と直線$x = \mathrm{x_array}[i+1]$に挟まれた区間スラブ$i$で辺$L^k$と交わる辺があるならば，6.へ，そうでないならば，7.へ．

6. L^k上で，zから一番近い交点z'を求め，$z := z'$とおき，11. へ．

7. 辺L^kと直線x = x_array[$i+1$]の交点z'を求め，$z := z', i := i+1$とし，11. へ．

8. もし直線x = x_array[$i-1$]と直線x = x_array[i]に挟まれた区間で辺L^kと交わる辺があるならば，9. へ，そうでないならば，10. へ．

9. L^k上でzから，一番近い交点z'を求め，$z := z'$とおき，11. へ．

10. 辺L^kと直線x = x_array[$i-1$]の交点z'を求め，$z := z', i := i-1$とし，11. へ．

11. zを，Zの端点として登録する．

12. $z \neq z_1$であるならば，3. へそうでないならば，終了する．

3. において，最も右側にある辺L^kを求める操作を行う際には，その直前に進んできた辺がどれで，どちらの方向 (右か左) に進んできたかという情報を用いる．具体的には，例えば，7. から 3. に戻ってきた場合には，直前の辺をL^kとすると，L^kは，右方向に進む辺で，zは，直線x = x_array[i]の上にある．L^kが垂直な辺でないとき，slab_array[i][j] = Re[z]となる辺L^jの中で，

1. edge[j].x_ℓ < x_array[i]を満たし，slab_array[$i-1$][j] < slab_array[$i-1$][k]を満たす辺 (L^kよりも下にある辺) があれば，その中で slab_array [$i-1$][j]が最大の辺L^jを左方向に進む辺として選び，それがない場合は，

2. 垂直な辺で edge[j].y_ℓ < Re[z] ≦ edge[j].y_rを満たす辺があれば，その中で edge[j].y_ℓが最小な辺L^jを左方向に進む辺として選び，それがない場合は，

3. edge[j].x_r > x_array[i]を満たす辺があれば，その中で slab_array[$i+1$][j]が最小な辺 (最も下にある辺) を右方向に進む辺として選び，それがない場合は，

4. 垂直な辺で edge[j].y_ℓ ≦ Re[z] < edge[j].y_rを満たす辺があれば，その中で edge[j].y_ℓが最大な辺L^jを右方向に進む辺として選び，それがない場合は，

5. edge[j].x_ℓ < x_array[i]を満たし，slab_array[$i-1$][j] > slab_array[$i-1$][k]を満たす辺 (L^kよりも上にある辺) があれば，その中で slab_array[$i-1$][j]が最大の辺L^jを左方向に進む辺として選ぶ．

なお，$z = z_1$である 1 番最初の段階では，直前の辺がないので，ここだけ別の処理が必要である．

また，6. から，3. に戻ってきたばあいは，zは，直線x = x_array[i]の上にないので，複数の辺が同一の点zで交わる場合には，それらの辺を全て記憶しておき，それらの中から上に述べたのと同様な方法で，最も右側にある辺L^kを求める必要がある．

上に示したアルゴリズムでは，簡単のため，2 次元配列 slab_array を用いて説明したが，slab_array[i][j] = ∞という要素は不要で，これを取り除かねば，スラブを用いる意味がない．スラブを用いる理由は，例えば，6. で一番近い交点z'を持つ辺L'を求めるという操作の際に，対象となる辺を制限したいためである．このためには，slab_array の情報を多重リストで実現すれば良い．ここでは，iが与えられた時に，jを変化させ，有限な値を持つ slab_array[i][j] を順に参照し，その際に，slab_array[$i-1$][j]，slab_array[$i+1$][j]，slab_array[i][j]，edge[j].y_ℓ，edge[j].y_r，edge[j].x_ℓ，edge[j].x_rの値を参照することができれば良いことを考慮すると，次のようなデータ構造を考えればよい．

```
type edge_pointer = ↑ edge;
type link = ↑ node;
node = record
        y: real;
        e_ptr: edge_pointer;
        node_next, x_prev, x_next: link
    end;
x_node = record
        x: real;
        top: link
    end;
```

として，大きさがrの x_node を要素とする 1 次元実数配列 x_node_array を準備し，有限な値を持つ slab_array [i][j] に相当するデータは，node.y に記憶することにする．iが与えられた時，slab_array[i][j] に対する参照は，x_node_array[i].top.を用いてリストをたぐり順に参照する．このとき，対応する node が見付かっているので，slab_array[$i-1$][j] や slab_array[$i+1$][j] は，node.x_prev↑.y node.x_next↑.y を参照すればよい．実際の応用においては，積の演算や逆集合の演算の結果短い辺が増加し，Z_kの辺の数はかなり多くなるが，半面，1 つのスラブに含まれる辺の数 (x_node[i].top で指されるリストの要素の数) はさほど多くない場合が多く，スラブを考える事は有効である．

上のアルゴリズムを用いることにより，NPIA の和の計算を行なうことができる．

積の演算については，辺と辺の積については，$C_L(L_X^i, L_Y^j)$という 2 次曲線を近似した部分が出てくるが，経験的には，多角形と多角形の積では，それらの$C_L(L_X^i,$

L_Y^j) という2次曲線を近似した部分で最終的な外部境界になるものが極めて少ないので，

$$U^{i,j} = \mathrm{outbd}[L_Z^i \odot w_q \bigcup L_Z^i \odot w_r \bigcup z_p \odot L_W^j \bigcup z_s \odot L_W^j] \quad (22)$$

$$U = \mathrm{outbd}[\bigcup_{L_Z^i \in E_Z,\, L_W^j \in E_W} U^{i,j}] \quad (23)$$

とおくとき，

$$Z \odot W = \mathrm{outbd}[U \bigcup (\bigcup_{L_Z^i \in E_Z,\, L_W^j \in E_W} L_Z^i \odot L_W^j)] \quad (24)$$

となることに注意して，まず，Uを先に述べたアルゴリズムを用いて計算し，その後，$C_L(L_X^i, L_Y^j)$ の端点で Uの外部にあるものがある場合には，多角形 Uと $L_Z^i \odot L_W^j$ の和集合を求めるという操作を繰り返した方が得策である．

また，逆集合の演算は，式(20)のように，$\{N_{Pinv}(L_Z^i)\ ; \varepsilon)\}$ を求めて，それらの和集合の外部境界を上のアルゴリズムを用いて求めてもよいが，実は，もっと簡単に求める方法がある．

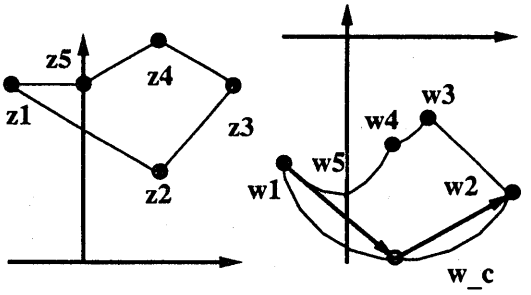

図 2: Zと $Z^{\ominus}$

$N_{Pinv}(L_Z^p);\ \varepsilon)$ を構成する3角形の端点の内2つは，Γ^pの上の点であり，もう1つの端点は，他の2つの端点におけるΓ^pの接線の交点であるものとする．このとき，Γ^pが，$\mathbf{V}(z^{-1}; Z)$ の境界の凹な部分にあるならば，$\{N_{Pinv}(L_Z^p); \varepsilon)\}$ を構成している3角形のΓ^pの上の点である2端点を結ぶ辺が，外部境界の一部となる．また，Γ^pが，$\mathbf{V}(z^{-1}; Z)$ の境界の凸な部分にあるならば，$\{N_{Pinv}(L_Z^p); \varepsilon)\}$ を構成している3角形のΓ^pの上の点である2端点と他の1点を結ぶ2つの辺が，外部境界の一部となる．なお，補題2のA)のようにΓ^pが元々線分になるばあいは，それ自体を外部境界の辺として選べばよい．従って，このようにして選んだ辺を順次つないで行けば，$Z^{\ominus}$が得られる．問題は，Γ^pが，$\mathbf{V}(z^{-1}; Z)$ の境界の凹な部分にあるか凸な部分にあるかということをどのようにして判定するかということである．Zの端点は，反時計回りに並べられているので，上のようにして構成された $Z^{\ominus}$の端点も反時計回りになっていることに注意する(図2を参照のこと)．$Z^{\ominus}$の端点を反時計回りに端点を訪れるとき，Γ^p の両端の点を w_p, w_sの順で訪れるとする．Γ^pの両端ではない点 w_cとすると，Γ^pが凸な部分であるならば，$w_s - w_c$は，$w_c - w_p$ の右側になければならないし，Γ^pが凸な部分であるならば，$w_s - w_c$は，$w_c - w_p$の左側になければならない．従って，$w_s - w_c$が，$w_c - w_p$のどちら側にあるか判定すれば，Γ^pが凸な部分であるか凹な部分であるか判定することができる．

Ackermann ら [6] は，(多角形とは限らない) 一般の集合に対して和および積を定義し，実際のインプリメントに関しては，従来は，画像処理的な方法を用いることを提案していたが，画素数を固定した場合には，少し複雑な問題に対しては，計算精度を保証することが困難であり，また精度を保つために必要な画素数は指数関数的に増加するという問題点があった．ごく最近に，文献 [6] の共著者である Sieuel から得た情報によると，この方法では，事実上，少し複雑な問題を解けないことを認識し，対象とする集合を多角形の集合に制限するとともに，画像処理的な方法を用いることをやめ，計算幾何学的な方法を採用することに方針を変更したようである．

4 適用例

上で述べた方法により NPIA のインプリメントを行った．プログラムは g++(GNU C++) を用いて開発した．このプログラムは，多角形区間演算を用いたロバスト制御系の解析・設計を行うための CAD システム (例えば，文献 [7]) の一部となっている．

以下では，簡単な NPIA の実行例を示す．

例 1 図3に4角形 Zと3角形の積の例を示す．この例では，2次曲線を近似した部分 $C_L(L_X^i, L_Y^j)$ は，全て最初に求めた式 (24) の Uの部分に含まれている．この様に，2次曲線を近似した部分が最終的な外部境界には含まれない場合も実際的にはかなりある． ■

例 2 機械系の特性方程式の値集合を求める例を示す．我々の開発した CAD システムでは，次に示すようなテキストファイルに書かれた内容を解読して，計算木 (正確には，DAG) を構成し，それに基づいて計算するようになっている．前半の部分の $polygon\ s\ =\ (0,1.12);$

や *polygon k* = [1.0, 1.25]; 等は，不確かさを含むパラメータの定義である．なお，(0, 1.12) は，複素数を示し，[1.0, 1.25] は，実区間を示す．また，後半部分の *define h1* = $((m1*s+d1)*s+c1+1)$; 等は，計算する関数の表現を与えている．このように，適当な中間変数を導入して，最終的な表現を与えてもよい．また，*operate*; は，演算を実行することを要求する命令で，*show*(*F*); は，計算した結果を画面に表示する命令である．この場合は，*F*だけを表示するようになっているが，他の中間変数も同様に表示することができる．我々の開発した CAD システムでは，もっと他に多数の命令を定義しているが，その詳細は，省略する．

polygon s = (0, 1.12);
polygon k = [1.0, 1.25];
polygon m1 = 2.0;
polygon d1 = [0.5, 2];
polygon c1 = [1, 2];
polygon m2 = 3.0;
polygon d2 = [0.5, 2];
polygon c2 = [2, 4];
define h1 = $((m1*s+d1)*s+c1+1)$;
define h2 = $((m2*s+d2)*s+c2+1)$;
define F = $h1*h2+(k-1)$;
*operate*1;
show(*F*);

計算結果を図 4 に示す．図 4 において，(A) は，$h1$, (B) は，$h2$, (C) は，$h1*h2$ の計算の前半部分 (式 (24) の Uの部分の計算結果) を示している．また，(D) は，2 次曲線を近似した部分 $C_L(L_X^i, L_Y^j)$ を付け加えている過程を図示している．最後の (E) は，Fの計算過程を示している． ■

上で示した例は比較的簡単であるので，問題を生ぜず，最終結果を得ることができたが，複素周波数 s が非常に大きくなる場合や，逆集合の計算が途中に含まれている場合には，非常に長い辺と短い辺が混在し，スラブの幅が相対的な意味で非常に狭くなり (x_array[i] と x_array[$i+1$] が，浮動小数点ではほとんど差がなくなる)，その結果，辺 L^j と $L^{j'}$ の交差する点の近くでは，slab_array[$i-1$][j] と slab_array[$i-1$][j'] の値が，丸め誤差により正しい大小関係を保てなくなる場合が生ずる．この様な場合は，先に述べた方法によって，最も右側にある辺を探索するのに失敗し，本当は最も右でないを選ぶと，もう少し先に進んだ状態では，探索している辺が内部の辺となってしまう．このような状態になると，無限ループに入り込むことになる．このような現象を如何に回避するかということが，今後の課題である，

5 おわりに

本文では，従来の凸多角形の集合の上で定義された PIA による値集合の推定の精度を向上させる 1 つの方法として，非凸多角形を含む多角形の集合の上での多角形区間演算 NPIA の定義を示すとともに，そのインプリメントの方法を示した．比較的簡単な例に対しては，正常に動作し，その有効性が確かめられたが，複雑な問題に対しては，丸め誤差により，アルゴリズムが破綻してしまうことがあり，これを解決する必要がある．

参考文献

1) 太田，田川，羽根田: 多角形区間演算の改良，第 24 回制御理論シンポジュウム予行集，pp.15-18 (1995).

2) Y. Ohta, L. Gong and H. Haneda: Polygon interval arithmetic and design of robust control systems, *Proc. IEEE Conf. on Decision and Control*, Honolulu, Hawaii, pp.1065-1067 (1990).

3) 太田，安井，羽根田: 多角形区間演算を用いた不確かさをもつむだ時間制御系の安定度の解析，計測自動制御学会論文集，vol.27, no.6, pp.620-627 (1991).

4) Y. Ohta, A. Isogai, K. Tagawa and H. Haneda : Computation of almost exact gain margin by using polygon interval arithmetic, Proceedings of the First Asian Control Conference pp.177-180 (1994)

5) 太田，磯貝，田川，羽根田: 多角形区間演算を用いたゲイン余裕の算出，システム制御情報学会論文誌，vol.39, no.5 pp.212-221 (1995).

6) J. Ackermann: *Robust control systems with uncertain physical parameters*, Springer-Verlag, Berlin (1993).

7) Y. Ohta, F. Kawamura, A. Isogai, K. Tagawa and H. Haneda : Robust servosystem design by using PIA, Proceedings of the 1994 IEEE Industrial Electronics Conference

8) Y. Ohta, L. Gong and H. Haneda : Polygon interval arithmetic and interval evaluation of value sets of transfer functions, IEICE Trans. on Fundamentals of Electronics, Communications and Computer Science, vol.E77-A no.6 pp.1033-1042 (1994)

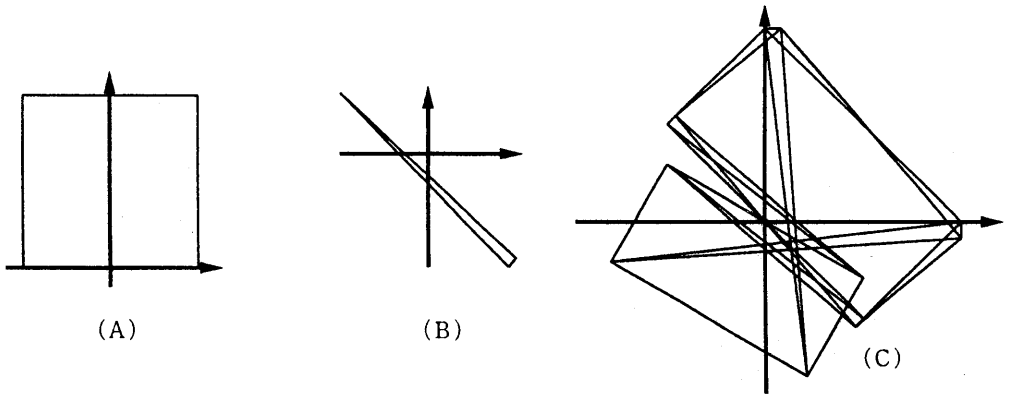

図3．多角形の積．　(A) Z, (B) W, (C) Z⊙W

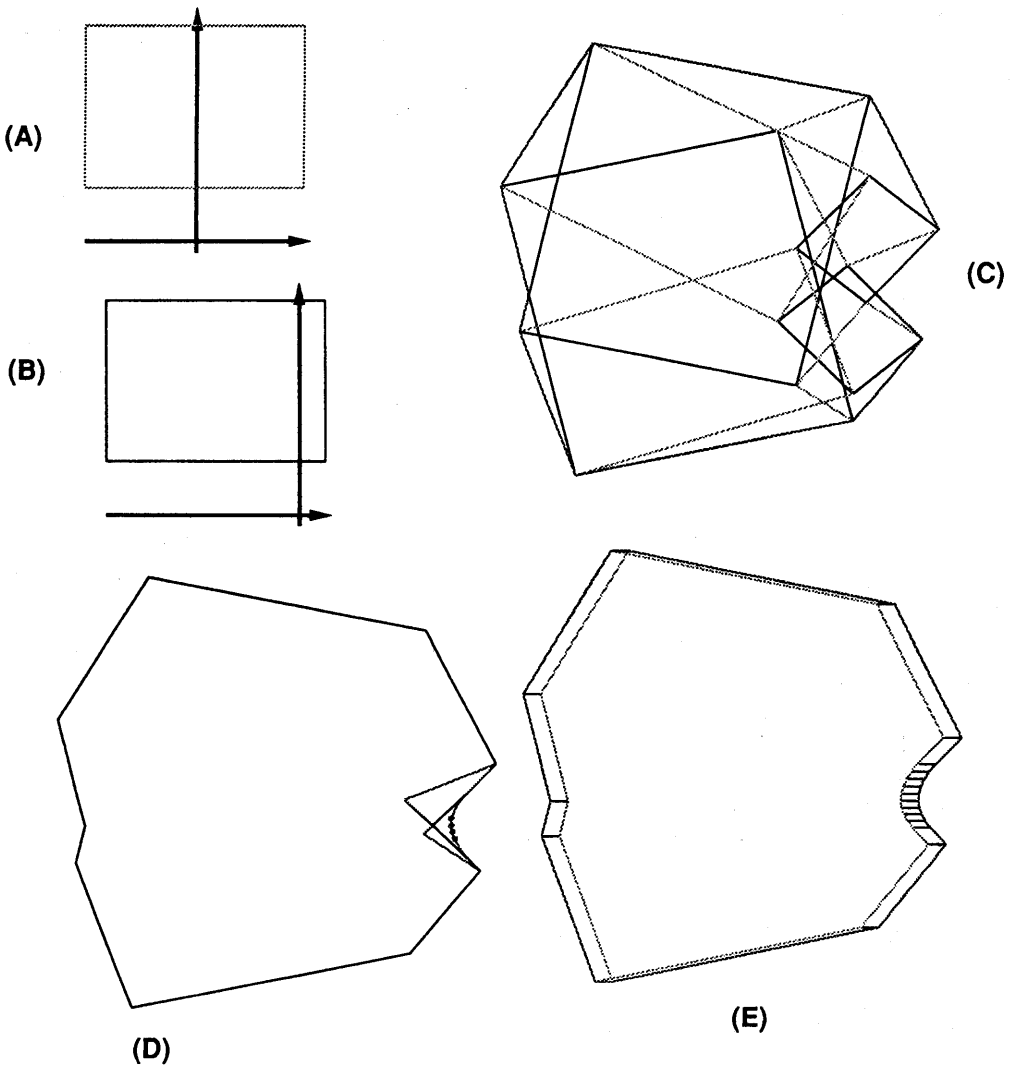

図4　例題2の計算結果