(Version: 2012/10/04)

Affine Arithmetic における ε の削減について

柏木 雅英

1 はじめに

数値計算において、計算を行うと同時にその結果の誤差評価をも同時に計算するような方法を総称して精度保証付き数値計算と呼び、近年急速な進歩を遂げている。精度保証付き数値計算の実現において最も基本的かつ重要な技法に、区間演算が挙げられる。区間演算とは、実数値を [下限, 上限] という 2 つの浮動小数点数で挟まれた区間で表現し、その区間同士の加減乗除等の演算を「演算結果として有り得る集合を包含するように」定義することにより行われるものである。そのとき、区間の両端を計算する際に丸めの向きを「外向き」にしておくことによって丸め誤差の影響分を区間内に収め、丸め誤差の把握を行うことが出来る。

区間演算の問題点の一つとして、確かに計算された区間は真の値を含むものの、区間幅が極端に広くなってしまうことが多い点が挙げられる。この現象は、区間演算が中間変数間の相関性を考慮していないために発生する。Affine Arithmetic は、中間変数間の相関性を考慮することによりこの問題点を解決する。後述するように、Affine Arithmetic では数値を複数のダミー変数 ε の線形結合の形で表現し、計算過程に乗算などの非線形演算が出現した場合は新たにダミー変数を付加することにより誤差を表現する。これにより複雑な関数の値域の評価を行えるが、それと引き換えにダミー変数 ε の数が増加し、それは計算時間の増大をもたらす。

本論文では、なるべく Affine Arithmetic の能力を損なうことなく、ダミー変数 ε の数を減らす手法を提案する。また、その性能を数値実験により示す。

2 Affine Arithmetic

2.1 Affine 形式

Affine Arithmetic は、変数間の相関性を考慮することにより区間演算の over-estimation の問題を解決する方法の一つである。この方法は文献 [1] で提案されたものである。また、文献 [2] の方法を簡略化したものと見ることも出来る。

Affine Arithmetic では、変動範囲が $-1 \leq \varepsilon_k \leq 1$ であるようなダミー変数 ε_k を用いて、その線形結合

$$a_0 + a_1\varepsilon_1 + \cdots + a_n\varepsilon_n$$

の形 (Affine 形式) で数 (区間) を表現する。計算機にはその係数 $a_0, \ldots, a_n$ を記憶する。なお、後述するように ε_k の最大数 n は計算の途中で変化する。

例えば、Affine 形式 x, y があって、それが

$$\begin{aligned} x &= 1 + 0.5\varepsilon_1 \\ y &= 1 + 0.5\varepsilon_2 \end{aligned}$$

であったとしよう。これは x と y に相関が無い状態で、このとき (x, y) が取り得る領域は図 1 のようになる。

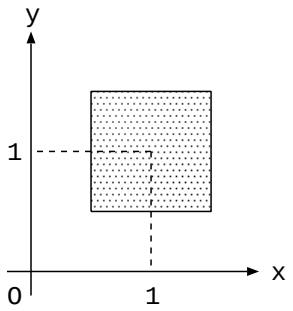

図 1: x と y に相関が無い場合

これに対して、

$$\begin{aligned} x &= 1 + 0.5\varepsilon_1 \\ y &= 1 + 0.4\varepsilon_1 + 0.1\varepsilon_2 \end{aligned}$$

では、x, y それぞれが取り得る範囲は $[0.5, 1.5]$ で変っていないが、ε_1 の係数を見ると分かるように両者には強い相関があり、(x, y) の取り得る領域は図 2 のようになる。

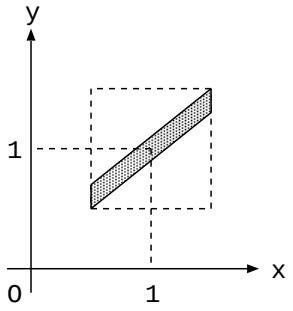

図 2: x と y に相関がある場合

また、この領域は、

$$\begin{aligned} x &= 1 + 0.5\varepsilon_1 \\ y &= 1 + 0.4\varepsilon_1 + 0.1\varepsilon_2 \end{aligned}$$

を

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 0.5 \\ 0.4 \end{pmatrix} \varepsilon_1 + \begin{pmatrix} 0 \\ 0.1 \end{pmatrix} \varepsilon_2$$

のような縦ベクトルの和に分解し、

- 点 $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$
- 係数ベクトル $\begin{pmatrix} 0.5 \\ 0.4 \end{pmatrix}$ を $-1 \sim 1$ 倍して得られる線分

- 係数ベクトル $\begin{pmatrix} 0 \\ 0.1 \end{pmatrix}$ を $-1 \sim 1$ 倍して得られる線分

のミンコフスキー和 (Minkowski sum) と考えることも出来る (図 3) 。

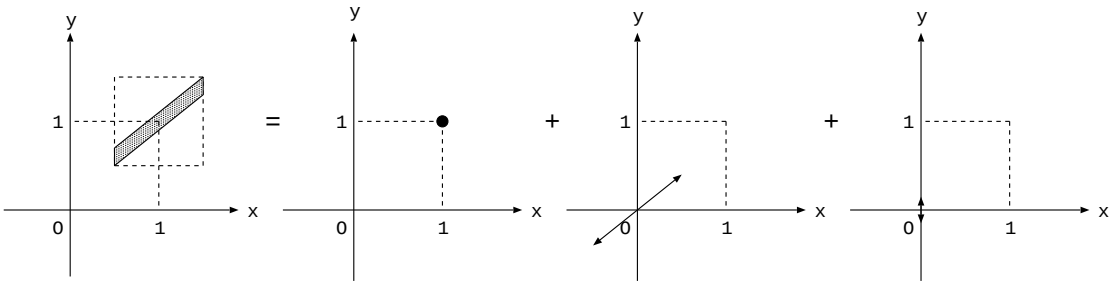

図 3: 集合のミンコフスキー和

n 変数関数 $f(x_1, \ldots, x_n)$ を Affine Arithmetic で評価する場合、最初に与えられる n 個の変数は一般に互いに無相関であるから、n 個の ε を用いて

$$\begin{aligned} x_1 &= \frac{\overline{x_1} + \underline{x_1}}{2} + \frac{\overline{x_1} - \underline{x_1}}{2}\varepsilon_1 \\ x_2 &= \frac{\overline{x_2} + \underline{x_2}}{2} + \frac{\overline{x_2} - \underline{x_2}}{2}\varepsilon_2 \\ &\vdots \\ x_n &= \frac{\overline{x_n} + \underline{x_n}}{2} + \frac{\overline{x_n} - \underline{x_n}}{2}\varepsilon_n \end{aligned}$$

のような affine 形式で初期化しておく。但し、入力変数 x_k の変域を $[\underline{x_k}, \overline{x_k}]$ とする。

なお、Affine 形式

$$x = a_0 + a_1\varepsilon_1 + \cdots + a_n\varepsilon_n$$

は、

$$[a_0 - \delta, a_0 + \delta], \quad \delta = \sum_{i=1}^{n} |a_i|$$

によっていつでも通常の区間に戻すことが出来る。

2.2 Affine 演算における丸め誤差

以下、Affine 形式同士の演算方法について述べる。ただし、まずは簡単のため、演算の過程で発生する丸め誤差を考慮しない形で説明することにする。例えば Affine 形式の係数が有理数表現されているならば、以下の説明の方法はそのまま実現出来る。

2.3 線形演算

Affine 形式の変数における演算は、加減算及び定数倍の場合は自明である。

Affine 多項式 x, y:

$$\begin{aligned} x &= x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n \\ y &= y_0 + y_1\varepsilon_1 + \cdots + y_n\varepsilon_n \quad , \end{aligned}$$

に対して、加算、減算は次のように定義する。

$$\begin{aligned} x \pm y &= (x_0 \pm y_0) + (x_1 \pm y_1)\varepsilon_1 + \cdots + (x_n \pm y_n)\varepsilon_n \\ x \pm \alpha &= (x_0 \pm \alpha) + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n \quad . \end{aligned}$$

定数の乗算は次のように定義する。

$$\alpha x = (\alpha x_0) + (\alpha x_1)\varepsilon_1 + \cdots + (\alpha x_n)\varepsilon_n \quad .$$

2.4 非線形単項演算

Affine 多項式

$$x = x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n$$

に対する非線形な演算 f について、$z = f(x)$ は一般に affine 多項式で表すことは出来ない。そこで、f を線形演算で近似し、近似誤差を新しいダミー変数 ε_{n+1} を導入することによって表すことを考える。

まず、x の変域 I を

$$I = [x_0 - \delta, x_0 + \delta], \quad \delta = \sum_{i=1}^{n} |x_i| \quad ,$$

で求める。次に、この領域 I において f をなるべくよく近似するような一次関数 $ax + b$ を求める (図 4 参照)。誤差の最大値

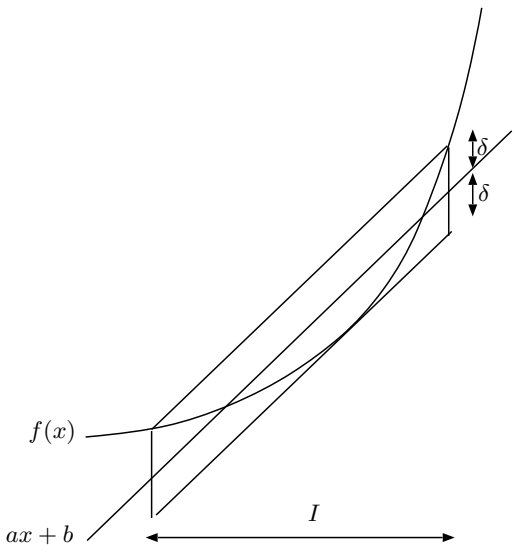

図 4: 非線形関数の線形近似

$$\delta = \max_{t \in I} |f(t) - (at + b)|$$

を求め、これをダミー変数 ε_{n+1} の係数とする。すなわち、非線形関数 f は区間 I において

$$f(x) \in ax + b + \delta\varepsilon_{n+1}$$

であり、よって単項演算の結果は

$$a(x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n) + b + \delta\varepsilon_{n+1}$$

とすればよい。これはダミー変数が一つ増加した Affine 形式である。このように定義された単項演算は、追加されたダミー変数の大きさが最小であるという意味で、最適である。

2.5 非線形二項演算

Affine 多項式 x, y:

$$\begin{aligned} x &= x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n \\ y &= y_0 + y_1\varepsilon_1 + \cdots + y_n\varepsilon_n \quad , \end{aligned}$$

に対する非線形な二項演算 $z = f(x, y)$ について考える。例えば、$x \times y$, x/y, x^y などである。

単項演算の場合と同様に、x と y の変域 X, Y を求め、$(x, y) \in X \times Y$ において $f(x, y)$ を最良近似する一次式 $ax + by + c$ を求め、$ax + by + c + \delta\varepsilon_{n+1}$ を結果とする方法を使うことは出来る。しかし、単項演算の場合と違ってこの方法は必ずしも最良の近似を与えるとは限らない。これは、x と y に相関性がある場合、点 (x, y) は一般に $X \times Y$ で与えられる長方形領域内の全ての点を取るとは限らないためである。

乗算の場合、変域 X, Y の半径を δ_x, δ_y とすると $x \times y$ は

$$y_0x + x_0y - x_0y_0$$

で最良近似され、誤差は $\delta_x\delta_y$ となるので、

$$\begin{aligned} z &= y_0x + x_0y - x_0y_0 + \delta_x\delta_y\varepsilon_{n+1} \\ &= x_0y_0 + \sum_{i=1}^{n}(y_0x_i + x_0y_i)\varepsilon_i \\ &\quad +(\sum_{i=1}^{n}|x_i|)(\sum_{i=1}^{n}|y_i|)\varepsilon_{n+1} \end{aligned}$$

が用いられることが多い。前述したように、これは最適では無い。

除算は、$x/y = x \times (1/y)$ のように分解し、逆数関数と乗算の組み合わせによって計算することが多い。これも最適では無い。

3 丸め誤差を考慮した Affine Arithmetic

計算機で実数をそのまま扱うことは出来ず浮動小数点数しか扱えないのが普通なので、前節のアルゴリズムをそのまま無誤差で実行することは出来ない。区間演算の場合は、単に「外側に」丸めることによって、真の変域を数学的に厳密に包含する性質を失うこと無く計算機上に実装することが出来る。しかし、Affine Arithmetic においては、ε の係数を上向きに丸めても下向きに丸めても厳密性が失われてしまう。

- (方法 1) 線形演算をも非線形演算と扱い、線形演算に対してもダミー変数を追加する。
- (方法 2) 丸め誤差を格納するための専用の ε を各中間変数に導入する。
- (方法 3) 非線形演算由来の誤差も、方法 2 の丸め誤差用 ε に押し込んでしまう。

などの方法がある。詳細は省略するが、まとめると、

	方法 1	方法 2	方法 3
線形計算における ε の追加	あり	なし	なし
非線形計算における ε の追加	あり	あり	なし
特殊ダミー変数 ε_r の使用	なし	あり	あり
包含性能	◎	○	△
計算速度	△	○	◎

のような違いがある。

4 ダミー変数 ε の削減

このように、Affine Arithmetic では計算が進行するにつれてダミー変数 ε の数が徐々に増えていき、それは計算時間の増大をもたらす。

本論文では、包含についての数学的厳密性を損なうこと無く、またなるべく過大評価が起きないように、ダミー変数 ε の数を削減する方法を示す。

大前提として、本論文で提案する ε の削減手法は、現在計算途中の全ての中間変数をなるべく多く同時に処理して初めて効果を発揮する。一部の少ない変数のみに対して適用しても大きな効果は得られない。

今、q 個のダミー変数 ε を持つ p 個の affine 変数

$$
\begin{aligned}
&a_{10} + a_{11}\varepsilon_1 + \cdots + a_{1q}\varepsilon_q \\
&a_{20} + a_{21}\varepsilon_1 + \cdots + a_{2q}\varepsilon_q \\
&\qquad\vdots \\
&a_{p0} + a_{p1}\varepsilon_1 + \cdots + a_{pq}\varepsilon_q
\end{aligned}
$$

を考える。

この affine 変数からいくつかの ε を選んで「区間化」してやれば、ε を消すことが出来る。S を消したい ε の添字集合として、

$$
\begin{aligned}
\sum_{i\in S} a_{1i}\varepsilon_i &\rightarrow (\sum_{i\in S} |a_{1i}|)\varepsilon_{q+1} \\
&\qquad\vdots \\
\sum_{i\in S} a_{pi}\varepsilon_i &\rightarrow (\sum_{i\in S} |a_{pi}|)\varepsilon_{q+p}
\end{aligned}
$$

と置き換えてやれば、添字 S を消すことが出来る。但し、新規に生成された区間を表すために p 個の新たな ε が追加される。

以下、ε を、r 個 $(p \leq r \leq q)$ に削減することを考える。ここで、$\varepsilon_1, \ldots, \varepsilon_q$ のうち、「区間化ペナルティの大きい」(具体的には後述) $r-p$ 個の ε を残し、区間化ペナルティの小さい $q-(r-p)$ 個を区間化することによって ε 数を削減することにする。すなわち、ペナルティの小さい $q-(r-p)$ 個を表す添字集合を S として、

$$\sum_{i \notin S} a_{1i}\varepsilon_i + (\sum_{i \in S} |a_{1i}|)\varepsilon_{q+1}$$
$$\vdots$$
$$\sum_{i \notin S} a_{pi}\varepsilon_i + (\sum_{i \in S} |a_{pi}|)\varepsilon_{q+p}$$

を計算結果とする。

以下、各 ε の区間化ペナルティの計算方法を示す。ベクトル $v_1, v_2, \ldots, v_q \in \boldsymbol{R}^p$ を

$$v_i = \begin{pmatrix} a_{i1} \\ \vdots \\ a_{ip} \end{pmatrix}$$

とする。

定義 1 (ペナルティ関数) ベクトル $v = \begin{pmatrix} a_1 \\ \vdots \\ a_p \end{pmatrix}$ に対して、ペナルティ関数 P を以下のように定義する:

- $a_1 = a_2 = \cdots = a_p = 0$ のとき、
$$P(v) = 0$$
- そうでないとき、a_i を絶対値が大きい順に並べたときの $1, 2$ 番目を a_s, a_t とする。すなわち、
$$|a_s| \geq |a_t| \geq |a_i| \quad (i \neq s, t)$$
が成り立つとする。$P(v)$ を、
$$P(v) = \frac{|a_s| \cdot |a_t|}{|a_s| + |a_t|}$$
で定める。

$v_1, \ldots, v_q$ から、$P(v_i)$ の値が小さいものを順に $q-(r-p)$ 個削減対象として選べばよい。

なお、この関数 P は、次の性質を満たす。

定理 1 (ペナルティ関数の性質) ベクトル $v = \begin{pmatrix} a_1 \\ \vdots \\ a_p \end{pmatrix} \in \boldsymbol{R}^p$ とし、$\boldsymbol{R}^p$ のノルムは最大値ノルムとする。ここで、

$$\begin{pmatrix} a_1 \\ \vdots \\ a_p \end{pmatrix} \varepsilon \quad (-1 \leq \varepsilon \leq 1)$$

で定められる $\boldsymbol{R}^p$ 内の線分を L 、

$$\begin{pmatrix} a_1\varepsilon_1 \\ \vdots \\ a_p\varepsilon_p \end{pmatrix} \quad (-1 \leq \varepsilon_i \leq 1)$$

で定められる $\boldsymbol{R}^p$ 内の超直方体領域をを B とすると、L と B のハウスドルフ距離は、

$$H(L, B) = 2P(v)$$

である。

つまり、$P(v)$ は元の v が表す線分 L と、それを区間化したときに生成される超直方体 B との最大距離を表しており (図 5)、これが小さいほど、区間化したときの領域の増大が小さいと言える。

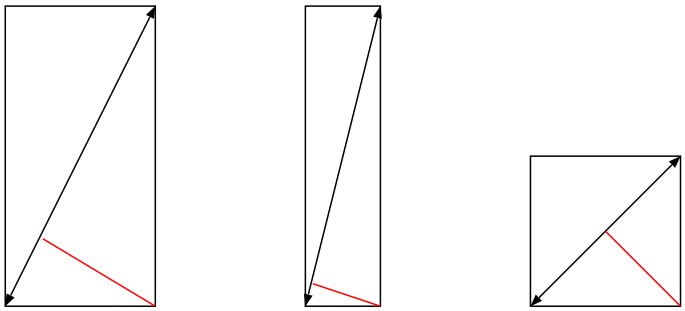

図 5: 線分と超直方体のハウスドルフ距離 (ペナルティ関数)

5 ε 削減の例

簡単な ε 削減の例を示す。

$$\begin{array}{rcccccccc} x & = & 1 & +\varepsilon_1 & -\varepsilon_2 & +0.1\varepsilon_3 & -0.3\varepsilon_4 & & +0.1\varepsilon_6 & +0.5\varepsilon_7 \\ y & = & 1 & +0.2\varepsilon_1 & +\varepsilon_2 & +0.05\varepsilon_3 & -0.3\varepsilon_4 & +0.5\varepsilon_5 & +0.03\varepsilon_6 & -0.2\varepsilon_7 \end{array}$$

という 7 つの ε を持った affine 変数 x, y を考える。点 (x, y) は $\boldsymbol{R}^2$ 内に図 6 のような集合を形成している。

$\varepsilon_1, \ldots, \varepsilon_7$ の係数ベクトル $v_1, \ldots, v_7$ に対するペナルティ関数の値は、

$$\begin{aligned} P(v_1) &= 1/6 = 0.1666\cdots \\ P(v_2) &= 1/2 = 0.5 \\ P(v_3) &= 1/30 = 0.0333\cdots \\ P(v_4) &= 3/20 = 0.15 \\ P(v_5) &= 0 \\ P(v_6) &= 3/130 = 0.0230\cdots \\ P(v_7) &= 1/7 = 0.142\cdots \end{aligned}$$

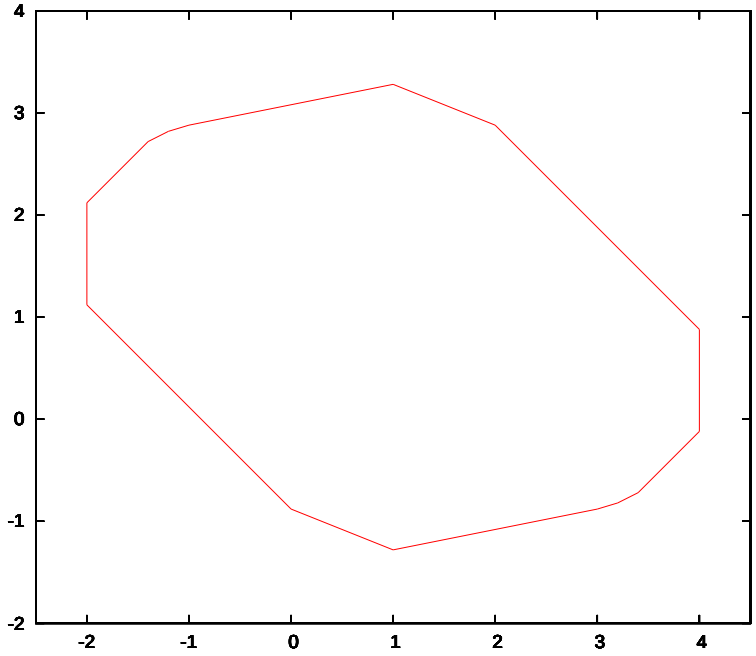

図 6: ε 削減前の集合 (ε を 7 個持つ)

ダミー変数を 5 個に削減する例を示す。ペナルティ関数の値を小さい方から $7-(5-2)=4$ 個選ぶ。すると、

- $\varepsilon_5, \varepsilon_6, \varepsilon_3, \varepsilon_7$ を区間化して消去し、
- $\varepsilon_1, \varepsilon_2, \varepsilon_4$ を残せばいい

ことが分かる。よって、

$$\begin{aligned}\begin{pmatrix} x \\ y \end{pmatrix} &= \begin{pmatrix} 1+1\varepsilon_1-\varepsilon_2-0.3\varepsilon_4+(|0.1|+|0|+|0.1|+|0.5|)\varepsilon_8 \\ 1+0.2\varepsilon_1+\varepsilon_2-0.3\varepsilon_4+(|0.05|+|0.5|+|0.03|+|0.2|)\varepsilon_9 \end{pmatrix} \\ &= \begin{pmatrix} 1+1\varepsilon_1-\varepsilon_2-0.3\varepsilon_4+0.7\varepsilon_8 \\ 1+0.2\varepsilon_1+\varepsilon_2-0.3\varepsilon_4+0.78\varepsilon_9 \end{pmatrix}\end{aligned}$$

のようになる。これを図示すると、図 7 のようになる。このように、あまり領域を大きくすること無く、ε を 7 個から 5 個に削減することが出来た。

なお、計算機上への affine arithmeric の実装の仕方によっては、

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1+1\varepsilon_1-\varepsilon_2-0.3\varepsilon_3+0.7\varepsilon_4 \\ 1+0.2\varepsilon_1+\varepsilon_2-0.3\varepsilon_3+0.78\varepsilon_5 \end{pmatrix}$$

のように ε の添字を付け替えて添字の最大数を減らさないと高速化出来ないこともある。この場合、ε の持つ意味が変わってしまうため、本アルゴリズムによる ε の削減を施された affine 変数と、そうでない affine 変数を混在して使用することは出来なくなる。

同様に、ε を 4 個に減らすと、

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1+1\varepsilon_1-\varepsilon_2+\varepsilon_8 \\ 1+0.2\varepsilon_1+\varepsilon_2+1.08\varepsilon_9 \end{pmatrix}$$

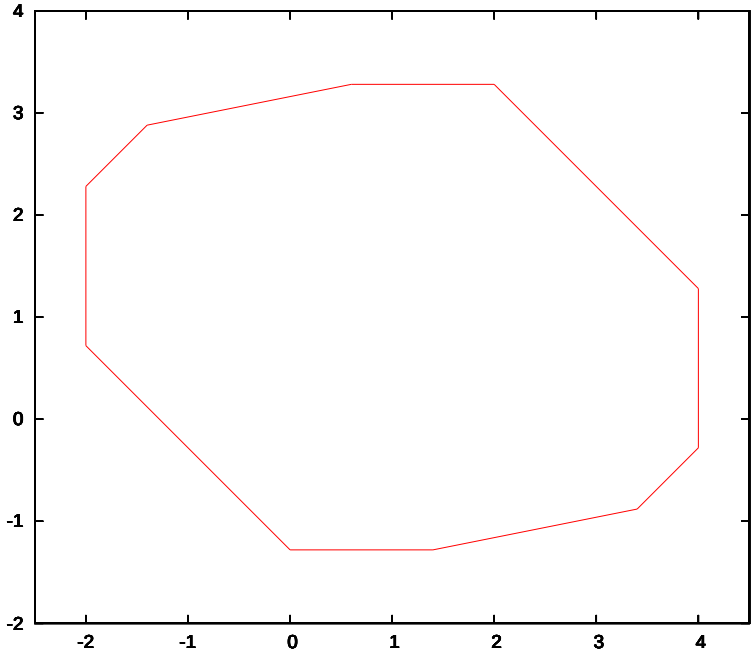

図 7: ε を 5 個に削減した集合

のようになる (図 8)。

ε を 3 個に減らすと、

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 - \varepsilon_2 + 2\varepsilon_8 \\ 1 + \varepsilon_2 + 1.28\varepsilon_9 \end{pmatrix}$$

のようになる (図 9)。

ε を 2 個に減らすのは単なる区間化と同じになり、

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 + 3\varepsilon_8 \\ 1 + 2.28\varepsilon_9 \end{pmatrix}$$

のようになる (図 10)。

以上を全て同時に図示すると、図 11 のようになる。

6 計算例 (Henon Map の計算)

Henon Map [3] と呼ばれる、chaotic な挙動で知られる 2 次元力学系

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = \begin{pmatrix} 1 - ax_i^2 + y_i \\ bx_i \end{pmatrix}$$

を例題とする。$b = 0.3$ のとき、$a \geq 1.06$ で chaos が発生することが知られている。ここでは、chaos 発生直前の $a = 1.05$ とし、区間幅の広がり方を調べる。

まず、初期値を

$$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} [-10^{-5}, 10^{-5}] \\ [-10^{-5}, 10^{-5}] \end{pmatrix}$$

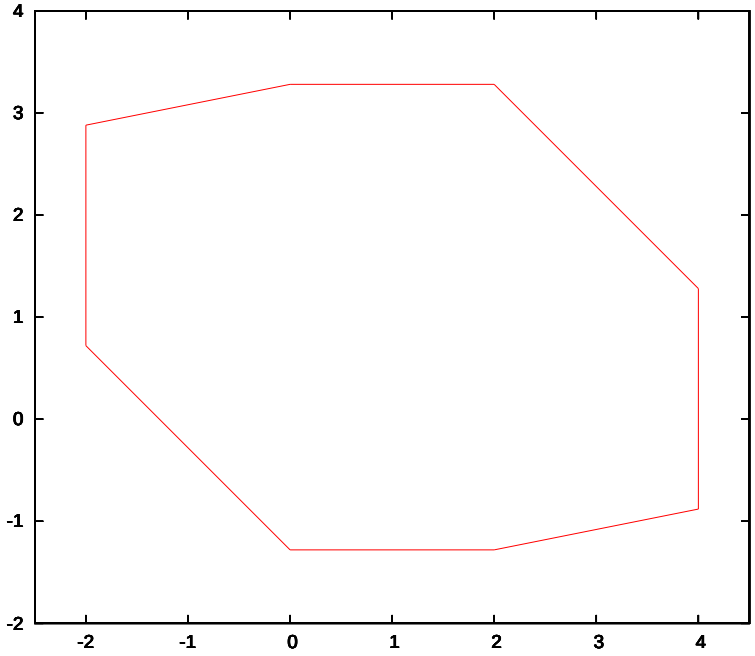

図 8: ε を 4 個に削減した集合

とし、軌道を

- 区間演算
- Affine Arithmetic (ε を減らさないオリジナル)
- Affine Arithmetic (ε の最大数 : 5 $\sim$ 15)
- Affine Arithmetic (ε の最大数 : 10 $\sim$ 20)
- Affine Arithmetic (ε の最大数 : 20 $\sim$ 30)
- Affine Arithmetic (ε の最大数 : 40 $\sim$ 50)

で計算した例を図 12 に示す。横軸は反復回数、縦軸は区間幅 (x, y のうち大きい方) である。なお、「ε の最大数 : $n \sim m$」は、「一回反復する度に ε の最大数をチェックし、それが m を超えていたら n まで削減する」ということを意味する。

これを見ると、ε の数の減少に伴って包含性能が低下してること、区間演算に比べれば高い性能を維持していることが分かる。

計算時間を、表 1 に示す。CPU は core i7 2640M (2.8G)、OS は ubuntu 10.04 LTS (64bit) である。

初期区間幅を 0 にして同様の計算を行った例も図 13 に示す。

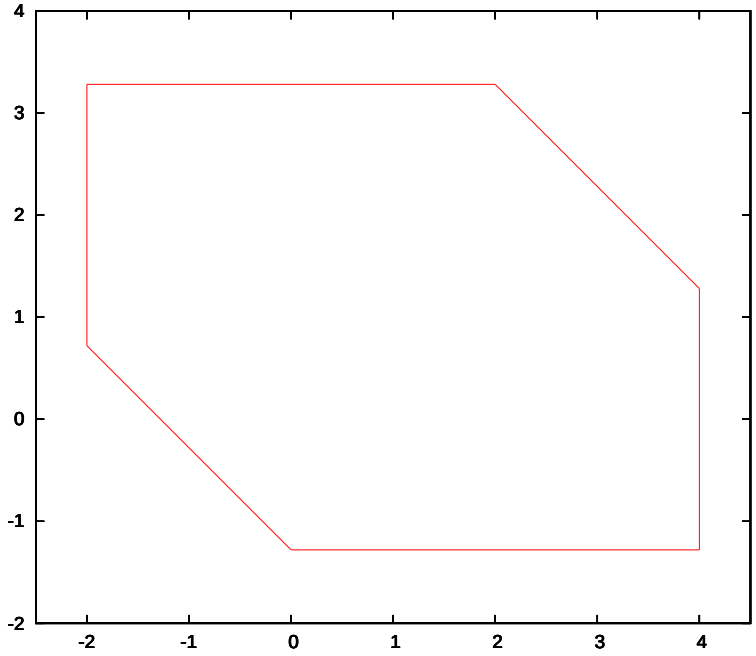

図 9: ε を 3 個に削減した集合

区間演算	0.15
Affine Arithmetic (オリジナル)	35.77
Affine Arithmetic (ε の最大数 5 $\sim$ 15)	2.1
Affine Arithmetic (ε の最大数 10 $\sim$ 20)	2.81
Affine Arithmetic (ε の最大数 20 $\sim$ 30)	3.71
Affine Arithmetic (ε の最大数 40 $\sim$ 50)	4.9

表 1: 計算時間 (単位: msec)

7 付録: ユークリッドノルムの場合

ベクトル $v = \begin{pmatrix} a_1 \\ \vdots \\ a_p \end{pmatrix} \in \boldsymbol{R}^p$ とし、$\boldsymbol{R}^p$ のノルムはユークリッドノルムとする。

このとき、区間化前後のハウスドルフ距離を与えるペナルティ関数 $P(v)$ は、次で与えられる。

$S = \{1, 2, \ldots, p\}$ を添字集合、2^S を S の部分集合全体の集合とすると、

$$P(v) = \sqrt{\frac{\max\limits_{S' \in 2^S} \left(\sum\limits_{i \in S'} a_i^2 \right) \left(\sum\limits_{i \in S - S'} a_i^2 \right)}{\sum\limits_{i \in S} a_i^2}}$$

となる。

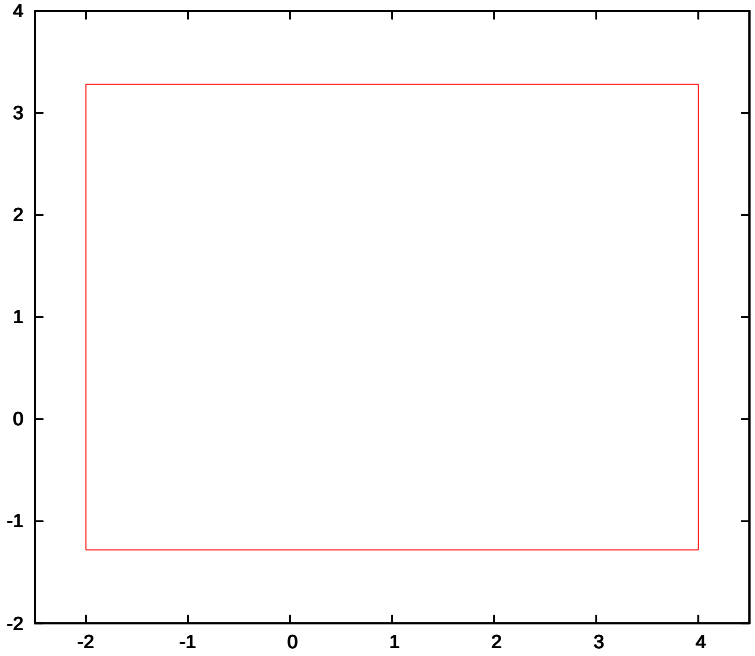

図 10: ε を 2 個に削減した集合

最大化するには、

$$\left(\sum_{i\in S'} a_i^2\right)\left(\sum_{i\in S-S'} a_i^2\right)$$

を最大化するように $a_1^2, a_2^2, \ldots, a_p^2$ を二つに分ければよい。すなわち、なるべく両集合の和が近くなるように二分する。この問題は、Number Partitioning Problem と呼ばれ、p が大きい場合は思いの外計算時間がかかる。単に捨てる ε の順序を決めるのに使うだけなので、計算しやすい最大値ノルムの方のペナルティ関数を使うのが良いと思われる。

参考文献

[1] Marcus Vinícius A. Andrade, João L. D. Comba and Jorge Stolfi: “Affine Arithmetic”, INTERVAL’94, St. petersburg (Russia), March 5-10, 1994.

[2] Masahide Kashiwagi : “Interval Arithmetic with Linear Programming — Extension of Yamamura’s Idea —”, Proc. 1996 International Symposium on Nonlinear Theory and its Applications (NOLTA’96 Symposium), pp.61–64 (Kochi, Japan, October 7–9, 1996).

[3] M. Henon : “A two-dimensional mapping with a strange attractor”, Communications in Mathematical Physics 50 (1), pp.69–77 (1976).

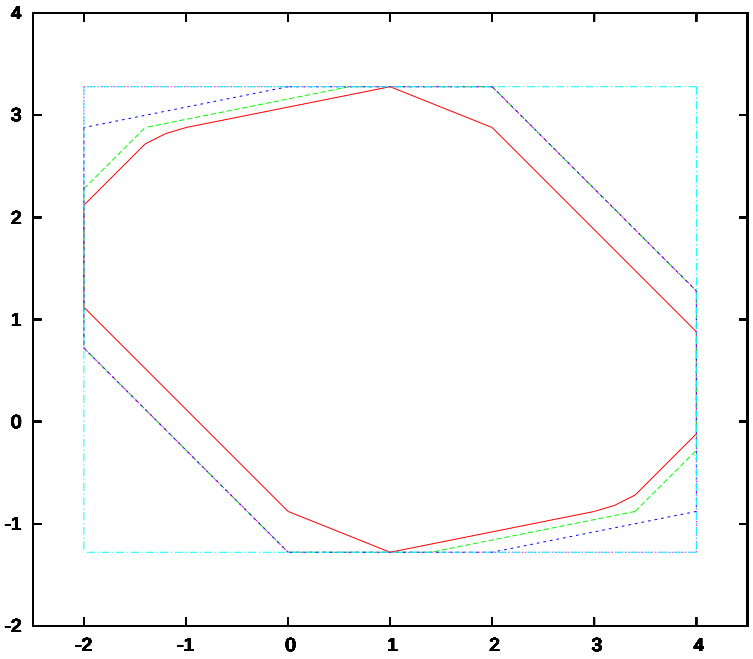

図 11: 全て同じ図に表示

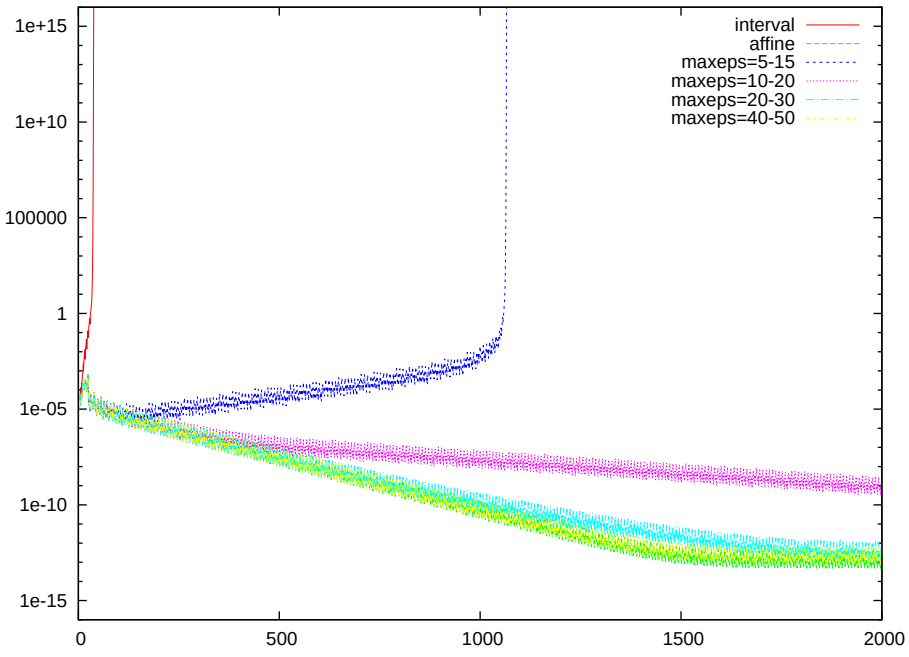

図 12: Henon Map の軌道の区間幅 (初期区間幅 $= 2 \times 10^{-5}$)

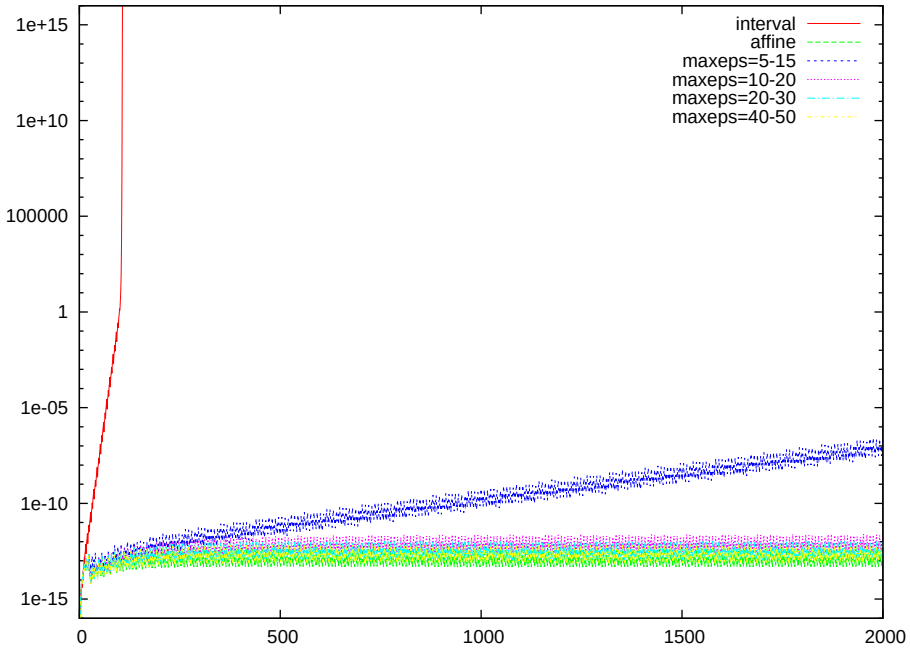

図 13: Henon Map の軌道の区間幅 (初期区間幅 = 0)