平成 10 年度

卒業論文

Affine Arithmetic の除算の実現

平成 11 年 2 月 5 日

指導教授： **柏木　雅英** 助教授

早稲田大学　理工学部　情報学科

g95p088-1 **長島 雄一郎**

目次

第 1 章

序

連続数学の問題は実数の体系の上で記述される。これを計算機で扱う場合、現代のコンピューター技術では実数を浮動小数点数で近似して計算するのだが、計算機で扱う浮動小数点数は有限個であるので、これだけでは連続数学の問題を扱うことができない。そこで2つの浮動小数点数を両端とする「区間」の中に連続数学の問題をの解を包み込み、その区間同士の加減乗除等の演算を「演算結果としてあり得る集合を包含するように」定義することにより行う、区間演算というアイデアが生まれた。

上のアイデアを生かすために、区間を数の拡張と考えることが始められた（幅の十分狭い区間に求めようとする実数が入ることが示されれば、実用上十分であるからである）。区間を数と考えると、その四則演算が定義されなければならない。それが区間演算である[1]。

この区間演算の大きな欠点として、計算された区間は確かに真の値を含むものの、区間幅が極端に大きくなってしまうことが挙げられる。たとえば、ここで簡単な例として$f(x) = x^2 - 2x + 1$を例にとって考えてみる。通常の区間演算の手法で区間$[0.9, 1.1]$において演算を行うと、$[-0.39, 0.41]$という結果になってしまう。正解は$[0, 0.01]$であるので、これは実に80倍の開きである。このような区間の増大は、x^2と$2x$が同じxの関数で互いに相関があるにも関わらず、区間演算における演算ではそれを無視して独立な値として計算を行ってしまうことに原因がある。

本卒業論文で取り上げるAffine Arithmeticは区間演算の一種であり、通常の区間演算でよく見られる区間の爆発的な増大が起こりにくいという特徴を持っている[2]。この手法は通常の区間演算に似ているが、一般の区間演算における区間が他の区間に全く影響しないのに対して、Affine Arithmeticでは変量が互いに関係しあっているという特徴を持つ。これによってAffine Arithmeticは計算式が膨大な場合、計算式が多くの変数を含んでいるような場合、またはcancellationによるエラーが生じるような場合に特に効果を発揮する。先の例をAffine Arithmeticで計算すると、表1.1のような結果になる。

特に注目すべきことは、区間$[0.9, 1.1]$において$[-0.01, 0.01]$という結果になることである。実はこの例は区間演算の特に苦手とするパターンではあるが、Affine Arithmeticの方が一般の区間演算と比べて高度な精度保証を備えているという一例にはなる。

しかしAffine Arithmeticは、非線形関数をどのように計算するかによってその精度が大

	区間演算	区間幅	Affine Arithmetic	区間幅
$[0.3, 0.5]$	$[0.09, 0.65]$	0.55	$[0.23, 0.49]$	0.26
$[0.5, 0.7]$	$[-0.15, 0.49]$	0.64	$[0.07, 0.25]$	0.18
$[0.7, 0.9]$	$[-0.31, 0.41]$	0.72	$[-0.01, 0.09]$	0.1
$[0.9, 1.1]$	$[-0.39, 0.41]$	0.8	$[-0.01, 0.01]$	0.02
$[1.1, 1.3]$	$[-0.39, 0.49]$	0.88	$[-0.01, 0.09]$	0.1
$[1.3, 1.5]$	$[-0.31, 0.65]$	0.96	$[0.07, 0.25]$	0.18
$[1.5, 1.7]$	$[-0.15, 0.89]$	1.04	$[0.23, 0.49]$	0.26

表 1.1: Affine Arithmetic の有効性を示す例

きく変化する。様々な単項演算を行う非線形関数（$\sin x$ や $1/x$ など）はすでに最良の計算方法が発見されているが、Affine 形式[1]どうしの乗算を行う方法は最良の方法がまだ見つかっていない。また、Affine 形式どうしの除算についてはまだほとんどその計算方法の研究はなされておらず、$x/y = x \times (1/y)$ というふうに、単項演算 $1/x$ と Affine 形式どうしの乗算を合成して行っている。本論文ではこの Affine 形式どうしの除算をいかに効率よく行うかを考察する。

---

[1] 第 3章を参照

第 2 章

区間演算

本章では Affine Arithmetic の解説への前座として、区間演算の説明を行う。なぜなら区間演算は数値計算のジャンルにおいてもっとも一般的な手法であり、さらに Affine Arithmetic の特徴は区間演算との比較で明確になるからである。

区間解析では区間を数の拡張と考える。そして、その四則演算が定義される。これを区間演算という。以下、区間演算について述べていく。

2.1 導入

区間解析においては、区間とは

$$[\underline{x}, \overline{x}] = \{x \in R \mid \underline{x} \le x \le \overline{x}\}$$

と表される閉区間であるとする。ただし、$\underline{x} \le \overline{x} \ \in R$で、それぞれの区間の下端、上端とする。以下、記号の節約のため、

$$[x] = [\underline{x}, \overline{x}]$$

と区間を表すことにし、区間 $[x]$ と単に書けば、それは $[x] = [\underline{x}, \overline{x}]$ を表すものとする。$\underline{x} = \overline{x}$ となる区間 $[x]$ は点区間という。点区間は実数であるので、点区間 $[x]$ の表す実数を x と書くことにする。

区間 $[x]$ について以下を定義する。

$$\begin{aligned} d([x]) &= \overline{x} - \underline{x} \\ r([x]) &= \frac{\overline{x} - \underline{x}}{2} \\ m([x]) &= \frac{\overline{x} + \underline{x}}{2} \end{aligned}$$

$d([x]), r([x]), m([x])$ を、それぞれ区間 $[x]$ の直径、半径、中心という。

2.2 区間演算

2 つの区間 $[x],[y]$ が与えられたときその 2 つの区間の四則演算を次で定義する。

$$[x]\circ[y]=\{x\circ y \mid x\in[x], y\in[y]\}$$
$$(\text{ただし}\circ\in\{+,-,\times,/\})$$

これを区間演算という。この定義では区間演算は無限回の実数演算をしないと実行できないような印象を与えるが、次が成立する。

$$[x]+[y]=[\underline{x}+\underline{y},\overline{x}+\overline{y}]$$
$$[x]-[y]=[\underline{x}-\overline{y},\overline{x}-\underline{y}]$$
$$[x]\times[y]=[\min\{\underline{x}\underline{y},\overline{xy},\underline{x}\overline{y},\overline{x}\underline{y}\},\max\{\underline{x}\underline{y},\overline{xy},\underline{x}\overline{y},\overline{x}\underline{y}\}]$$
$$[x]/[y]=[x]\times\left[\frac{1}{\overline{y}},\frac{1}{\underline{y}}\right],\ (0\notin[y])$$

これから区間演算が有限回の実数演算で実行できることがわかる。乗算（表 2.1）と除算（表 2.2）においては、場合分けにより、より少ない手間で計算する次のような計算規則も簡単に導ける。

	$y\geq 0$	$y\ni 0$	$y\leq 0$
$x\geq 0$	$[\underline{x}\underline{y},\overline{xy}]$	$[\overline{x}\underline{y},\overline{xy}]$	$[\overline{x}\underline{y},\underline{x}\overline{y}]$
$x\ni 0$	$[\underline{x}\overline{y},\overline{xy}]$	$[\min\{\underline{x}\overline{y},\overline{x}\underline{y}\},\max\{\underline{x}\underline{y},\overline{xy}\}]$	$[\overline{x}\underline{y},\underline{x}\underline{y}]$
$x\leq 0$	$[\underline{x}\overline{y},\overline{x}\underline{y}]$	$[\underline{x}\overline{y},\underline{x}\underline{y}]$	$[\overline{xy},\underline{x}\underline{y}]$

表 2.1: 区間 $[x],[y]$ の乗算 $[x][y]$

区間演算については、包含関係における単調性

$$[x]\subseteq[x'],\ [y]\subseteq[y']\rightarrow[x]\circ[y]\subseteq[x']\circ[y'],\ \circ\in\{+,-,\times,/\}$$

が成立する。

	$y \geq 0$	$y \leq 0$
$x \geq 0$	$[\underline{x}/\overline{y}, \overline{x}\underline{y}]$	$[\overline{x}/\overline{y}, \underline{x}\underline{y}]$
$x \ni 0$	$[\underline{x}/\underline{y}, \overline{x}\underline{y}]$	$[\overline{x}/\overline{y}, \underline{x}\overline{y}]$
$x \leq 0$	$[\underline{x}/\underline{y}, \overline{xy}]$	$[\overline{x}/\underline{y}, \underline{x}\overline{y}]$

表 2.2: 区間 $[x], [y]$ の除算 $[x]/[y]$

また、加法と乗法に関し交換則と結合則が成立する。

$$[x] \circ [y] = [y] \circ [x], \ \circ \in \{+, \times\}$$

$$[x] \circ ([y] \circ [z]) = ([x] \circ [y]) \circ [z], \ \circ \in \{+, \times\}$$

しかし、加法と乗法の逆元は存在しない。すなわち、$-[x] = [-\overline{x}, -\underline{x}]$ であるが

$$0 = [0] \subseteq [x] - [x] = [\underline{x} - \overline{x}, \overline{x} - \underline{x}]$$

$$1 = [1] \subseteq [x]/[x]$$

となることがわかる。上式で等号は $[x]$ が点区間のときのみ成立する。

また、分配則も区間演算に対しては成立しない。そのかわり次の劣分配則が成立する。

$$[x] \times ([y] + [z]) \subseteq [x] \times [y] + [x] \times [z]$$

この式で等号は例えば区間 $[y]$ と $[z]$ が同じ符合をもつときに成立する。

2.3 むすび

本章では、区間演算の概説を行った。区間演算はその考え方が簡単で分かりやすいが、単純に区間演算を行っていくと、区間の幅が大きく広がってしまい、その結果の値を使おうとしても非常に扱いにくくなってしまう。それは区間が単純に演算可能になっているだけで、それぞれの変量の相関関係などの内容については全く考慮されていないからである。

Affine Arithmetic はこの区間演算の改良版であり、区間演算よりもよりタイトに演算を行うことができる。次章でその仕組みと演算の実装を紹介する。

第 3 章

Affine Arithmetic の導入

本性では Affine Arithmetic の概念と用語の説明を行い、また演算の定義を行っていく。

3.1 Affine 形式

Affine Arithmetic において通常の区間演算でいう「区間」は、次のような１次の多項式、Affine 形式（*Affine form*）で表される。

$$x = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + \cdots + x_n\varepsilon_n$$

ここで、x_i は実数であり、ε_i はそれが区間 $U = [-1, 1]$ に含まれることだけが分かっているような変数であり、x の変量部分のそれぞれ独立した構成要素である。

上の x_0 をこの Affine form の *central value*、係数 x_i を部分偏差（*partial deviations*）、ε_i を *noise symbols* という。noise symbol である ε_i は、それぞれ独立した値をとって変量 x を構成し、係数 x_i がそのふれ幅を決定している。

Affine 多項式は、通常の区間演算でいう「区間」に次のようにして変換できる。

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

ここで区間演算での「区間」と、Affine 形式で表される区間の違いを簡単な例を挙げて説明する。

以下のように x と y が Affine 形式で表されているとする。

$$\begin{array}{rcccccccc} x & = & 10 & + & 2\varepsilon_1 & + & 3\varepsilon_2 & & \\ y & = & 12 & - & 3\varepsilon_1 & & & + & 1\varepsilon_3 \end{array}$$

ここで x は 5 から 15 の値をとり、y は 8 から 16 の値をとることがすぐに分かる。しかし実際には x と y が共通する変量 ε_1 を持つため、x と y は全く独立に値をとるというわけではない。具体的には図 3.1のような範囲をとる。

ここで濃いグレーの部分が Affine Arithmetic で表される範囲で、薄いグレーと濃いグレーの両方で表される領域が一般の区間演算で表される範囲である。つまり、この場合は薄いグレーの分だけ Affine Arithmetic は一般の区間演算よりタイトに演算できるのである。

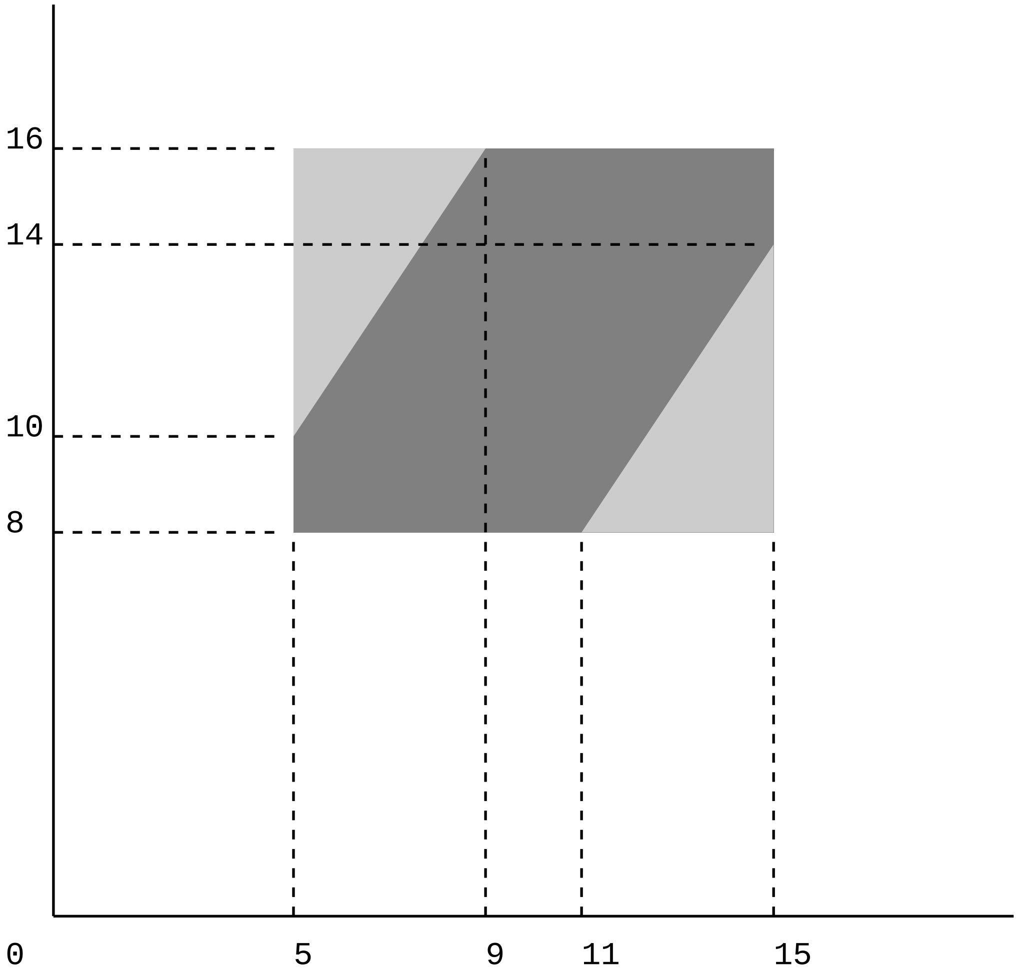

図 3.1: AA と IA の区間

3.2　一般的な演算の定義

2 つの Affine 多項式 x, y

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

に対して、以下に各種演算の定義を行なう。演算とは

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

という $f(x, y)$ を

$$z = z_0 + z_1\varepsilon_1 + \cdots + z_n\varepsilon_n$$

のように、Affine 多項式の形を満たすように表すことである。

各種 Affine 演算は次のように定義できるが、それは線形なものとそうでない場合に分けることができる。ここでは前者を Affine 演算 (*Affine-Operation*)、後者を非 Affine 演算 (*Non-Affine-Operation*) と呼ぶことにする。

3.3　Affine 演算

演算 f が線形な場合の演算は、以下のように定義できる。

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

定数倍も次のように定義できる。

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

3.4　非 Affine 演算

演算 f が非線形な場合の演算を考える。次の式のように、z を ε の式で表すことを考える。

$$\begin{aligned} z &= f(x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n,\ y_0 + y_1\varepsilon_1 + \cdots + y_n\varepsilon_n) \\ &= f^*(\varepsilon_1, \cdots, \varepsilon_n) \end{aligned}$$

この場合は一般的に z を正しく表すような $z_0, \cdots, z_n$ を決定することはできない。よってつぎのような f^a を与え、

$$f^a(\varepsilon_1, \cdots, \varepsilon_n) = z_0 + z_1\varepsilon_1 + \cdots + z_n\varepsilon_n$$

f^* と f^a の誤差を $z_{n+1}\varepsilon_{n+1}$ として次のように表すことができる。

$$\begin{aligned} z &= f^a(\varepsilon_1, \cdots, \varepsilon_n) + z_{n+1}\varepsilon_{n+1} \\ &= z_0 + z_1\varepsilon_1 + \cdots + z_n\varepsilon_n + z_{n+1}\varepsilon_{n+1} \end{aligned}$$

ここで z_{n+1} とは f^* と f^a の誤差の最大値であるから

$$|z_{n+1}| \geqq \max |f^*(\varepsilon_1, \cdots, \varepsilon_n) - f^a(\varepsilon_1, \cdots, \varepsilon_n)|$$

である。

3.4.1　単項演算の定義

$f(x) = \sin x$ や $f(x) = e^x$ などの単項演算については、一般に曲線 $y = f(x)$ を x_0 の付近で直線 $y = ax + b$ の形に近似し、その時の最大誤差を δ として次のように計算できる。

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

ここで δ は関数 $f(x)$ と直線 $y = ax + b$ の区間 $[\underline{x}, \overline{x}]$ における差の最大値であるから

$$\delta = \max |f(t) - (at + b)|$$

であり、等号成立の時に最良の近似を与える。つまり直線 $y = ax + b$ どのように設定し、δ をいかに小さな値にするかが Affine の単項演算の焦点となる。

3.4.2　単項演算の実際

以下に具体的な例として、x を Affine 形式として、

$$\begin{aligned} f(x) &= \frac{1}{x} \\ &= \frac{1}{x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n} \end{aligned}$$

の、区間 $[\underline{x}, \overline{x}]$ における最適な近似を求める。つまり図 3.2のように $f(x) = 1/x$ を区間 $[\underline{x}, \overline{x}]$ において直線 $y = ax + b$ で近似することを考える。

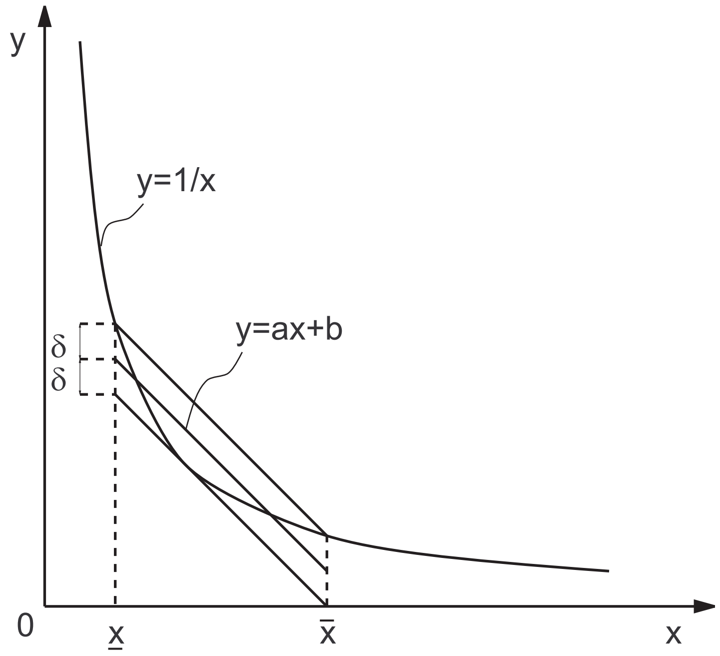

図 3.2: 関数 $y = \frac{1}{x}$ の近似

求める方針としては図形的直感を活かし、以下のような方針で行うことにする。

1. 図 3.2の「上の直線」を求める。この直線が区間 $[\underline{x}, \overline{x}]$ の最大値となっている。
2. 同様に「下の直線」を求める。こちらは区間 $[\underline{x}, \overline{x}]$ の最小値となっている。
3. 直線 $y = ax + b$ を求める。これは最大値と最小値の真ん中に来るようにする。そうすれば誤差 δ を最小にできるからである。

まず、上限を表す直線を求める。点 $(\underline{x}, f(\underline{x}))$ $(\overline{x}, f(\overline{x}))$ を通るので、

$$\begin{aligned} y &= \frac{f(\overline{x}) - f(\underline{x})}{\overline{x} - \underline{x}}(x - \underline{x}) + f(\underline{x}) \\ &= -\frac{1}{\underline{x}\overline{x}}x + \frac{\underline{x} + \overline{x}}{\underline{x}\overline{x}} \end{aligned}$$

となる。

次に下限を表す直線を求める。それは上限を表す直線と傾きが同じで、かつ関数 $y = f(x) = 1/x$ と接する直線である。$f'(x) = -\frac{1}{x^2}$ を解き、接点 $x = \sqrt{\underline{x}\overline{x}}$ を得るので

$$y = -\frac{1}{\underline{x}\overline{x}}x + \frac{2}{\sqrt{\underline{x}\overline{x}}}$$

となる。

最後に、求める直線は以上の 2 つの直線の真ん中であるので、

$$y = -\frac{1}{\underline{x}\overline{x}}x + \frac{\underline{x} + \overline{x}}{2\underline{x}\overline{x}} + \frac{1}{\underline{x}\overline{x}}$$

を得る。

以上により非線形関数 $f(x) = 1/x$ を $[\underline{x}, \overline{x}]$ において最適な近似を与える直線が求まった。この直線と実際に関数 $f(x)$ がとる幅との誤差 δ を評価する。

$$\delta = \max|f(x) - (\text{近似式})| = (\text{上限の式}) - (\text{近似式})$$

であるから、

$$\delta = \frac{\underline{x} + \overline{x}}{2\underline{x}\overline{x}} - \frac{1}{\sqrt{\underline{x}\overline{x}}}$$

となる。

以上により関数 $f(x) = 1/x$ は

$$-\frac{1}{\underline{x}\overline{x}}x + \frac{\underline{x} + \overline{x}}{2\underline{x}\overline{x}} + \frac{1}{\sqrt{\underline{x}\overline{x}}} + \delta\varepsilon_{n+1}$$

として計算できることが分かった。

3.4.3　乗算の近似

x, y を Affine 形式とする。ここで $z = f(x, y) = xy$ を考える。単純に x, y を展開し、乗算をすると以下のような形になる。

$$\begin{aligned} x \times y &= (x_0 + \sum_{i=1}^{n} x_i\varepsilon_i) \times (y_0 + \sum_{i=1}^{n} y_i\varepsilon_i) \\ &= x_0y_0 + \sum_{i=1}^{n}(x_0y_i + y_0x_i)\varepsilon_i + (\sum_{i=1}^{n} x_i\varepsilon_i) \times (\sum_{i=1}^{n} y_i\varepsilon_i) \\ &= f^a(\varepsilon_1, \cdots, \varepsilon_n) + (\sum_{i=1}^{n} x_i\varepsilon_i) \times (\sum_{i=1}^{n} y_i\varepsilon_i) \end{aligned}$$

ここで最後の項は ε どうしの積の形がでてしまい、Affine 形式とならない。そこでこの項全体を新たに $\delta\varepsilon_{n+1}$ として計算する。

δ の値は単純に ε がその振れ幅の最大値をとるとして

$$\begin{aligned} &\max\left|(\sum_{i=1}^{n} x_i\varepsilon_i) \times (\sum_{i=1}^{n} y_i\varepsilon_i)\right| \\ &= (\sum_{i=1}^{n} |x_i|) \times (\sum_{i=1}^{n} |y_i|) = \delta \end{aligned}$$

として大きめに計算する。よって乗算を計算するには

$$x_0y_0 + \sum_{i=1}^{n}(x_0y_i + y_0x_i)\varepsilon_i + \delta\varepsilon_{n+1}$$

を行えばよい。

このδの値は、先ほど大きめにとってしまったので必ずしも常に最良の値をとるわけではないが、かならず真の値を包含し、また計算しやすいので現時点では Affine 形式どうしの乗算の主流となっている。

3.4.4　除算の近似

いまのところ Affine 変量どうしの除算は、第 3.4.2節で紹介した単項演算 $f(x) = 1/x$ と第 3.4.3節で紹介した Affine 形式どうしの乗算を組み合わせて行う方法

$$\frac{x}{y} = x \times \left(\frac{1}{y}\right)$$

以外は知られていない。

この方法は1回の計算で近似を2回行い、従って計算の量が増えてしまうだけでなく、1回の計算でεが2つ増えてしまいロスが大きくなってしまう。

以降の章でこの除算を1回の近似で行う方法を考察していく。

第 4 章

除算の近似方法

前章で述べたように Affine 形式どうしの除算は、近似的な式を作り、変量 δ を評価し、ε を増やさなければ計算することができない。そこで x/y を Affine 多項式の形である $ax+by+c$ で近似することにする。xyz 平面において曲面 $z=x/y$ を、平面 $z=ax+by+c$ で、ある区間 $[\underline{x},\overline{x}][\underline{y},\overline{y}]$ で図形的に近似することを考える。

4.1 近似の方針

計算をなるべく簡単にするため a,b,c があまり複雑にならないようにする、ということに重点をおいて、平面は以下の条件を満たすように設定することにした。

その概要は

- 直線 $y=y_0$を含む
- 点 (x_0,y_0) での曲面の y軸方向の傾きをもつ

そしてその平面 $z=ax+by+c$ と平面 $z=x/y$の誤差を

$$\delta=\max_{\substack{\underline{x}\leqq x\leqq\overline{x}\\ \underline{y}\leqq y\leqq\overline{y}}}\left|\frac{x}{y}-(ax+by+c)\right|$$

で求め、を評価することにした。ただし、$x_0=\dfrac{\underline{x}+\overline{x}}{2}$、$y_0=\dfrac{\underline{y}+\overline{y}}{2}$とする。

1. 直線 $y=y_0$を含むならば

   $y=y_0$が曲面と平面の式の両方の式で成り立つので

   $$\frac{x}{y_0}=ax+by_0+c$$

   これが x,yにたいして恒等的に成り立つので

   $$a=\tfrac{1}{y_0}$$
   $$by_0+c=0$$

   となる。

2. 点 (x_0, y_0) での曲面の傾きをもつならば

(x_0, y_0) での y軸方向の傾きは$-\frac{x_0}{{y_0}^2}$なので

$$b = -\frac{x_0}{{y_0}^2}$$

となる。また、$by_0 + c = 0$ より

$$c = \frac{x_0}{y_0}$$

となる。

以上により求めたい平面の式は

$$z = \frac{1}{y_0}x - \frac{x_0}{{y_0}^2}y + \frac{x_0}{y_0}$$

と分かった。

4.2 誤差評価

この近似の誤差の値は

$$\left| \frac{x}{y} - \left\{ \frac{x}{y_0} - \frac{x_0}{{y_0}^2}y + \frac{x_0}{y_0} \right\} \right|$$

で与えられ、さらにこれを変形し見やすくすると

$$\left| (y - y_0) \left\{ -\frac{x}{yy_0} + \frac{x_0}{{y_0}^2} \right\} \right|$$

を得る。これの最大 / 最小についても調べる。

最大最小をとりうる時の x は $x = \underline{x}$ または $x = \overline{x}$ のいずれかであるので（下図参照）、x をその 2 つのいずれかに限定し、上の式を y についての関数と見て評価する。

yについての式として書き直すと

$$f(y) = \frac{x_0}{{y_0}^2}y + \frac{x}{y} + \frac{-x - \frac{x_0}{{y_0}^2}}{y_0}$$

であり、最大最小を調べると $y = \underline{y}, y = y_0\sqrt{\frac{x}{x_0}}, y = \overline{y}$ のいずれかで最大値最小値をとることが分かった。

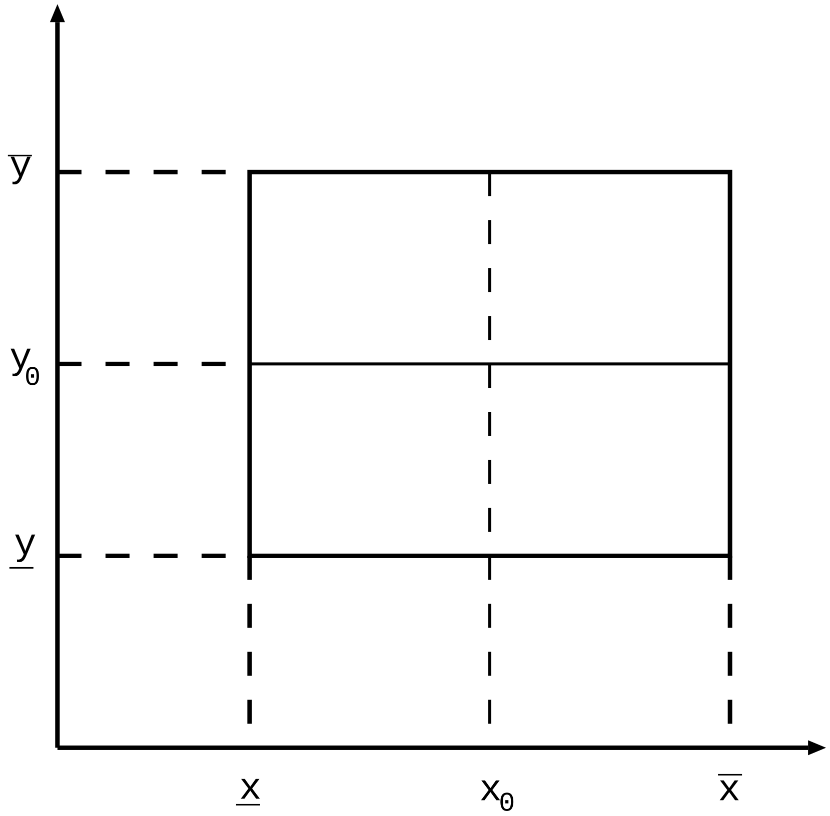

図 4.1: $z = x/y$ を上から見た図

つまり、次の 6 つの点を調べれば最大値を調べられることがわかった。(ただし $\underline{x}, x_0$ と $\overline{x}, x_0$ のいずれかが異符号となる場合は $\sqrt{\ }$ の中を計算しなくてよいので 5 点となる)。

$$\begin{array}{ccc} (\underline{x}, \underline{y}) & (\underline{x}, \overline{y}) & (\underline{x}, y_0\sqrt{\frac{\underline{x}}{x_0}}) \\ (\overline{x}, \underline{y}) & (\overline{x}, \overline{y}) & (\overline{x}, y_0\sqrt{\frac{\overline{x}}{x_0}}) \end{array}$$

よって、このときの $|f(y)|$ の最大値をδとすると、

$$\frac{x}{y} = \frac{1}{y_0}x - \frac{x_0}{{y_0}^2}y + \frac{x_0}{y_0} + \delta\varepsilon_{n+1}$$

となる。

以上のようにすれば Affine 形式どうしの除算を 1 回の近似で行えることが分かった。

4.3 むすび

本章で Affine 形式どうしの除算を 1 回の計算で行う方法を考案し、解説した。次章でこの手法の有効性を実際に計算機上でプログラムを走らせて検証する。

第 5 章

数値実験

本章では、第 4 章で考案したアルゴリズムを実際にプログラミング言語で実行した結果を掲載し、その有効性を検証する。

5.1 数値実験

以下にプログラムを走らせた時の実行ログと、それをまとめたものを掲載する。本論文で考察した除算を 1 回の近似で行う方法 (x/y) と、2 回の近似に分けて行う方法 ($x \times (1/y)$) を同じデータで計算した。なお、本論文では丸め誤差は考慮していないため、それを気にしなくてもよい `calc` という言語で行った。

5.1.1 変量が少ない場合

変量が小さい場合をまず計算してみた。

次のような p, q

$$p = 5 + 0.5\varepsilon_1$$

$$q = 10 + 0.1\varepsilon_2$$

を計算したところ、次のような結果になった。

```
> read affine

> read input

> p

      5+.05e1

> q

      10+0e1+.1e2

> q/p

      2+-.02e1+0e2+~.00040404040404040404e3

> affine_range(q/p)

      [+~1.97959595959595959595, +~2.02040404040404040404]
```

```
> q*(1/p)
        ~2.00010000750062505469+~-.02000200020002000200e1
        +~.02000100007500625054e2+0e3+
~.00010001250137514532e4+~.00020102012701395147e5
> affine_range(q*(1/p))
        [+~1.95979597459720970534, +~2.04040404040404040404]
>
```

これをまとめると以下のような結果になった。

$$q/p = 2 - 0.02\varepsilon_1 + 0.00\dot{0}\dot{4}\varepsilon_3$$

$$q \times (1/p) = 2 - 0.02\dot{0}00\dot{2}\varepsilon_1 + 0.02\varepsilon_2 + 0.0001\varepsilon_4 + 0.0002\varepsilon_5$$

q/p の区間は $[1.97\dot{9}\dot{5}, 2.02\dot{0}\dot{4}]$ であり、$q \times (1/p)$ の区間は $[1.9598, 2.\dot{0}\dot{4}]$ となった。つまりこの場合は q/p の方がいい結果が出ることが分かった。

5.1.2 分子と分母の間に共通する ε がない場合

次に、多くの ε を持ち、互いに共通する変量がない場合を試してみた。

次のような x, y

$$x = 137.525 + 0.1\varepsilon_1 + 0.5\varepsilon_2 + 0.56\varepsilon_3 + 3.015\varepsilon_4 + 0.66\varepsilon_5$$

$$+0.625\varepsilon_6 + 1.03\varepsilon_7 + 0.295\varepsilon_8 + 0.51\varepsilon_9 + 0.62\varepsilon_{10} + 1.5\varepsilon_{11}$$

$$y = 88.41 + 0.555\varepsilon 12 + 0.555\varepsilon_{13} + 2.94\varepsilon_{14} + 0.1\varepsilon_{15} + 0.05\varepsilon_{16}$$

$$+0.1\varepsilon_{17} + 1.715\varepsilon_{18} + 1.5\varepsilon_{19} + 2.94\varepsilon_{20} + 0.16\varepsilon_{21} + 9.145\varepsilon_{22} + 4.46\varepsilon_{23}$$

を計算したところ、次のような結果になった。

```
> read input
```

```
> x

    137.525+.1e1+.5e2+.56e3+3.015e4+.66e5

    +.625e6+1.03e7+.295e8+.51e9+.62e10+1.5e11

> y

    94.68+0e1+0e2+0e3+0e4+0e5+0e6+0e7+0e8+0e9

    +0e10+0e11+.555e12+.555e13+2.94e14+.1e15

    +.05e16+.1e17+1.715e18+1.5e19++.16e21+9.145e22+4.46e23

> x/y

    ~1.45252429235318969159+~.00105618926911702577e1

    +~.00528094634558512885e2+~.00591465990705534431e3

    +~.03184410646387832699e4+~.00697084917617237008e5

    +~.00660118293198141106e6+~.01087874947190536544e7

    +~.00311575834389522602e8+~.00538656527249683143e9

    +~.00654837346852555978e10+~.01584283903675538656e11

    +~-.00851448016746958469e12+~-.00851448016746958469e13

    +~-.04510373277902807027e14+~-.00153414057071524048e15

    +~-.00076707028535762024e16+~-.00153414057071524048e17

    +~-.02631051078776637432e18+~-.02301210856072860728e19

    +~-.04510373277902807027e20+~-.00245462491314438477e21

    +~-.14029715519190874239e22+0e23+~.16190518842537032578e24

> affine_range(x/y)

    [+~.88803270746711986955, +~2.01701587723925951363]

> x*(1/y)

    ~1.50251704299830637452+~.00109254102381262052e1

    +~.00546270511906310261e2+~.00611822973335067492e3

    +~.03294011186795050877e4+~.00721077075716329545e5

    +~.00682838139882887826e6+~.01125317254526999138e7

    +~.00322299602024723054e8+~.00557195922144436466e9

    +~.00677375434763824724e10+~.01638811535718930784e11
```

```
        +~-.0091106663719157O854e12+~-.0091106663719157O854e13
        +~-.04826190834852645608e14+~-.00164156150845328081e15
        +~-.00082078075422664040e16+~-.00164156150845328081e17
        +~-.02815277986997376605e18+~-.02462342262679921228e19
        +~-.04826190834852645608e20+~-.00262649841352524931e21
        +~-.15012079994805253092e22+~-.07321364327701632453e23
        +0e24+~.05171339320525990517e25+~.03075917575025012290e26
> affine_range(x*(1/y))
        [+~.91959553930345350976, +~2.08543854669315923928]
```

これをまとめると以下のような結果になった。

$$
\begin{aligned}
x/y = {} & 1.4525243 + 0.0010562\varepsilon_1 + 0.0052809\varepsilon_2 + 0.00591475\varepsilon_3 \\
& +0.0318441\varepsilon_4 + 0.0069708\varepsilon_5 + 0.0066012\varepsilon_6 + 0.0108787\varepsilon_7 \\
& +0.0031158\varepsilon_8 + 0.0053866\varepsilon_9 + 0.0065484\varepsilon_{10} + 0.0158428\varepsilon_{11} \\
& -0.0085145\varepsilon_{12} - 0.0085145\varepsilon_{13} - 0.0451037\varepsilon_{14} - .001534141\varepsilon_{15} \\
& -0.0007670\varepsilon_{16} - 0.0015341\varepsilon_{17} - 0.0263105\varepsilon_{18} - 0.0230121\varepsilon_{19} \\
& -0.045103733\varepsilon_{20} - 0.0024546\varepsilon_{21} - 0.1402972\varepsilon_{22} + 0.1619052\varepsilon_{24} \\
x \times (1/y) = {} & 1.5025170 + 0.0010925\varepsilon_1 + 0.0054627\varepsilon_2 + 0.0061182\varepsilon_3 \\
& +0.0329401\varepsilon_4 + 0.0072108\varepsilon_5 + 0.0068283\varepsilon_6 + 0.0112532\varepsilon_7 \\
& +0.0032230\varepsilon_8 + 0.0055720\varepsilon_9 + 0.0067738\varepsilon_{10} + 0.0163881\varepsilon_{11} \\
& -0.0091110\varepsilon_{12} - 0.0091107\varepsilon_{13} - 0.0482619\varepsilon_{14} - 0.0016416\varepsilon_{15} \\
& -0.0008208\varepsilon_{16} - 0.0016416\varepsilon_{17} - 0.0281528\varepsilon_{18} - 0.0246234\varepsilon_{19} \\
& -0.0482619\varepsilon_{20} - 0.0026265\varepsilon_{21} - 0.1501208\varepsilon_{22} - 0.0732136\varepsilon_{23} \\
& +0.0517134\varepsilon_{25} + 0.0307592\varepsilon_{26}
\end{aligned}
$$

x/y の区間は $[0.8880327, 2.0170159]$ であり、$x \times (1/y)$ の区間は $[0.9195955, 2.0854385]$ と

なった。つまりこの場合もx/yの方がいい結果であることが分かった。

5.1.3 分子と分母の間に共通する ε が多く含まれている場合

最後に、互いに共通する変量 ε を多く持つ場合を試してみた。次のような x, y

$$
\begin{aligned}
x &= 105.88 + 0.1\varepsilon_1 + 0.5\varepsilon_2 + 0.56\varepsilon_3 \\
&\quad +3.015\varepsilon_4 + 0.05\varepsilon_{16} + 0.1\varepsilon_{17} + 1.715\varepsilon_{18} + 1.5\varepsilon_{19} \\
y &= 66.34 + 0.1\varepsilon_1 + 0.5\varepsilon_2 - 0.56\varepsilon_3 \\
&\quad +3.015\varepsilon_4 - 0.05\varepsilon_{16} - 0.1\varepsilon_{17} + 1.715\varepsilon_{18} + 1.5\varepsilon_{19}
\end{aligned}
$$

を計算したところ、次のような結果になった。

```
> x
        105.88+.1e1+.5e2+.56e3+3.015e4+0e5+0e6+0e7
        +0e8+0e9+0e10+0e11+0e12+0e13+0e14+0e15+.05e16
        +.1e17+1.715e18+1.5e19
> y
        66.34+.1e1+.5e2+-.56e3+3.015e4+0e5+0e6+0e7
        +0e8+0e9+0e10+0e11+0e12+0e13+0e14+0e15+-.05e16
        +-.1e17+1.715e18+1.5e19
> affine_range(x)
        [+98.34, +113.42]
> affine_range(y)
        [+58.8, +73.88]
> x/y
        ~1.59602050045221585770+~-.00089843307273472393e1
        +~-.00449216536367361966e2+~.02191395056155020922e3
        +~-.02708775714295192660e4+0e5+0e6+0e7+0e8+0e9+0e10
        +0e11+0e12+0e13+0e14+0e15+~.00195660272870984010e16
```

```
        +~.00391320545741968021e17+~-.01540812719740051546e18
        +0e19+0e20+0e21+0e22+0e23+~.03783537268419059387e24
> affine_range(x/y)
        [+~1.48251488624358474859, +~1.70952611466084696680]
> x*(1/y)
        ~1.60643007486187867287+~-.00092008642855956029e1
        +~-.00460043214279780149e2+~.02214532196598287762e3
        +~-.02774060582107074300e4+0e5+0e6+0e7+0e8+0e9+0e10
        +0e11+0e12+0e13+0e14+0e15+~.00197726088981989978e16
        +~.00395452177963979957e17+~-.01577948224979645912e18
        +~-.01380129642839340448e19+0e20+0e21+0e22+0e23+0e24
       +0e25+~.01047746779784899729e26+~.01383307978128178200e27
> affine_range(x*(1/y))
        [+~1.49120051957668734818, +~1.72165963014706999756]
```

これをまとめると以下のような結果になった。

$$\begin{aligned} x/y = {} & 1.5960205 - 0.0008984\varepsilon_1 - 0.0044922\varepsilon_2 + 0.0219140\varepsilon_3 \\ & -0.0270878\varepsilon_4 + 0.0019566\varepsilon_{16} + 0.0039132\varepsilon_{17} - 0.0154081\varepsilon_{18} \\ & +0.0378354\varepsilon_{24} \\ x \times (1/y) = {} & 1.6064301 - 0.0009201\varepsilon_1 - 0.0046004\varepsilon_2 + 0.0221453\varepsilon_3 \\ & -0.0277406\varepsilon_4 + 0.0019773\varepsilon_{16} + 0.0039545\varepsilon_{17} - 0.0157795\varepsilon_{18} \\ & -0.0138013\varepsilon_{19} + 0.0104775\varepsilon_{26} + 0.0138331\varepsilon_{27} \end{aligned}$$

x/y の区間は $[1.4825149, 1.7095261]$ であり、$x \times (1/y)$ の区間は $[1.4912005, 1.7216596]$ となった。

つまりこの場合も x/y の方がいい結果であることが分かった。

5.2　考察

ここで数値実験したものについてはすべて以前アルゴリズムである $x \times (1/y)$ の方法よりもいい精度で計算することができた。

これで『従来の方式である $x \times (1/y)$ よりもいい精度の除算のアルゴリズムを実現する」という目的を達成することができた。

第 6 章

まとめ

本論文では第2章で数値計算の分野でもっとも一般的なものである区間演算の解説を行ない、第3章でAffine Arithmeticの実装と各主演算の定義を行なった。それらを通じてAffine Arithmeticの有効性を検証してきた。

そして第4章でAffine Arithmeticの除算のアルゴリズムを解説し、その手法で計算すれば、わずかながらであるが、従来の除算の唯一の手法であった $x \times (1/y)$ と計算する方法よりも高い精度で計算することが可能になったということが第5章でわかった。また、ε の数を押える効果もあるので計算機資源を有効に使えるというメリットもある。

Affine Arithmeticは非線形関数の演算の方法が比較的難しいが、本論文で紹介したように研究次第ではより有効な演算方法が見つかる可能性が高い。Affine Arithmeticの有効性をもっと知ってもらい、さらなる研究がなされることを望む。

第 7 章

謝辞

本研究を進めるに当たり常に正しい道を示し、終始丁寧なご指導及びご激励を賜わり、その他多くの面でもいろいろとご面倒を見て下さり御助言を与えて下さいました柏木雅英助教授に深く感謝致します。

また、卒業論文中間報告の際など機会のあるごとに御指導、御鞭撻を賜わり、研究に方向性を与えて下さった大石進一教授に深く感謝致します。

また、中間発表前夜に深夜まで我々学生に付き添って拙い研究報告書を指導して下さった、大石研究室 情報学科助手である神沢雄智氏に深く感謝します。

また、日常生活においていろいろとお世話になりました、大石研究室修士課程2年 青木康裕氏、大熊伸也氏、大上勝博氏、西田貴洋氏、諸林操氏、並びに大石研究室修士課程1年 大野徹雄に深く感謝致します。

また、同じ柏木研究室の唯一の同期の学生であり、CG班でありながら助言をして下さいました柏木研究室4年 高崎大輔氏に深く感謝致します。

また、同じ精度保証班、非線形班として意見の交換や協力などをして下さいました大石研究室4年 佐藤仁一氏、清水吉直氏、紫村仁史氏、立夽重紀氏、藤本政信氏、並びに渡辺利明氏深く感謝致します。

最後に、卒業論文で数値計算野分野をやりたいという私の願いを聞き届けて下さり、快く柏木先生を紹介して下さった筧克彦教授に深く感謝します。

参考文献

[1] 大石進一:“非線形解析入門”，(現代非線形科学シリーズ１）, コロナ社, 1997.

[2] Marcus Vinícius Alvim Andrade, João Luiz Dihl Comba and Jorge Stolfi: “Affine Arithmetic”, INTERVAL’94, St. petersburg (Russia), March 5-10, 1994.

[3] R.E.Moore:“Interval Analysis”,Prentice-Hall, 1966.

[4] R.E.Moore:“Method and Applications of Interval Analysis”, SIAM, Philadelphia,1979.