区間演算

柏木 雅英
kashi@waseda.jp
http://verifiedby.me/

早稲田大学 基幹理工学部 応用数理学科

区間演算とは

区間演算

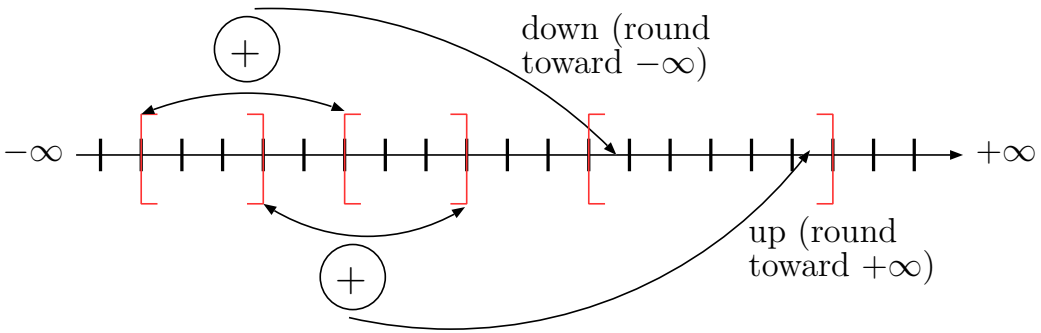
- 精度保証付き数値計算の基本技術。
- 例えば、円周率を $\pi \simeq 3.14$ とするとそれは近似だが、$\pi \in [3.14, 3.15]$ というのは正しい情報。
- 数値を、計算機で表現可能な浮動小数点数を両端に持つ閉区間 $X = [a, b]$ で表現する。
- 区間同士の演算は、集合値演算として計算結果として有り得る値を包含するように行う。
- IEEE754 標準で定義されている「方向付き丸め」を利用する。
- 「丸め誤差の把握」と「関数の値域の評価」の 2 つの役割がある。

丸めの向きを制御する

IEEE754 の丸めモード

- round to nearest: デフォルトの丸めモード。最も近い浮動小数点数に丸める。
- round toward $+\infty$: 上向き丸め
- round toward $-\infty$: 下向き丸め
- round toward 0: 絶対値が大きくならない方向への丸め

区間演算では、下端を下向き丸め、上端を上向き丸めで計算することによって、計算結果が外側に広がるように計算する。



加減乗除と平方根の計算ルール

- $X = [a, b], Y = [c, d]$
- $\underline{\cdot}, \overline{\cdot}$ はそれぞれ下向き丸め、上向き丸め

- 加算 $X + Y = [a \underline{+} c, b \overline{+} d]$
- 減算 $X - Y = [a \underline{-} d, b \overline{-} c]$
- 乗算 $X \times Y =$

	$d < 0$	$c \leq 0, d \geq 0$	$c > 0$
$b < 0$	$[b \underline{\times} d, a \overline{\times} c]$	$[a \underline{\times} d, a \overline{\times} c]$	$[a \underline{\times} d, b \overline{\times} c]$
$a \leq 0, b \geq 0$	$[b \underline{\times} c, a \overline{\times} c]$	$[\min(a \underline{\times} d, b \underline{\times} c), \max(a \overline{\times} c, b \overline{\times} d)]$	$[a \underline{\times} d, b \overline{\times} d]$
$a > 0$	$[b \underline{\times} c, a \overline{\times} d]$	$[b \underline{\times} c, b \overline{\times} d]$	$[a \underline{\times} c, b \overline{\times} d]$

- 除算 $X/Y =$

	$d < 0$	$c > 0$
$b < 0$	$[b \underline{/} c, a \overline{/} d]$	$[a \underline{/} c, b \overline{/} d]$
$a \leq 0, b \geq 0$	$[b \underline{/} d, a \overline{/} d]$	$[a \underline{/} c, b \overline{/} c]$
$a > 0$	$[b \underline{/} d, a \overline{/} c]$	$[a \underline{/} d, b \overline{/} c]$

($Y \not\ni 0$ の場合のみ定義される)

- 平方根 $\sqrt{X} = [\underline{\sqrt{a}}, \overline{\sqrt{b}}]$

丸め誤差の把握ができる

10 進数有効数字 3 桁で $(1 \div 3) \times 3$ を計算すると $\cdots$

- 普通の数値計算

$$1 \div 3 = 0.333$$
$$0.333 \times 3 = 0.999$$

四捨五入による誤差がでる。

- 区間演算

最初は幅のない区間からスタート。

$$[1, 1] \div [3, 3] = [0.333, 0.334]$$
$$[0.333, 0.334] \times [3, 3] = [0.999, 1.01]$$

常に区間内に真の値を含みながら計算が進む。

2 次方程式の丸め誤差

2 次方程式

2 次方程式 $ax^2 + bx + c = 0$ の解の公式は、

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

大きな誤差が見られる例

$$a = 1,\ b = 10^{15},\ c = 10^{14}$$

のときの大きい方の解を計算する。

- (解の公式) $\dfrac{-b + \sqrt{b^2 - 4ac}}{2a} = -0.125$
- (解の公式の分子を有理化)
  $\dfrac{2c}{-b - \sqrt{b^2 - 4ac}} = -0.10000000000000002$

2 次方程式の例を区間演算で計算する

2 次方程式

2 次方程式 $ax^2 + bx + c = 0$ の解の公式は、

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

大きな誤差が見られる例

$$a = 1,\ b = 10^{15},\ c = 10^{14}$$

のときの大きい方の解を計算する。

- (解の公式) $\dfrac{-b + \sqrt{b^2 - 4ac}}{2a} = [-0.1875, -0.0625]$
- (解の公式の分子を有理化) $\dfrac{2c}{-b - \sqrt{b^2 - 4ac}} =$
  $[-0.10000000000000004, -0.09999999999999991]$

丸めの向きの変更方法

丸めの向きの変更方法

- IEEE754 で丸めの向きを変更可能にすることが要請されているが、その方法は CPU やコンパイラによって違う。
- X86 だと FPU と SSE2 の 2 種類の演算器で丸めの向きの変更方法が異なるなど複雑。
- 最近は C99 準拠のコンパイラが増えたので、`fenv.h` と `fesetround` を使えばハードウェアの違いを吸収してくれる。

```
#include <iostream>
#include <fenv.h>
int main()
{
    double x=1, y=10, z;
    std::cout.precision(17);

    fesetround(FE_TONEAREST);
    z = x / y;
    std::cout << z << std::endl;
    fesetround(FE_DOWNWARD);
    z = x / y;
    std::cout << z << std::endl;
    fesetround(FE_UPWARD);
    z = x / y;
    std::cout << z << std::endl;
}
```

```
0.10000000000000001
0.099999999999999991
0.10000000000000001
```

実装上の注意

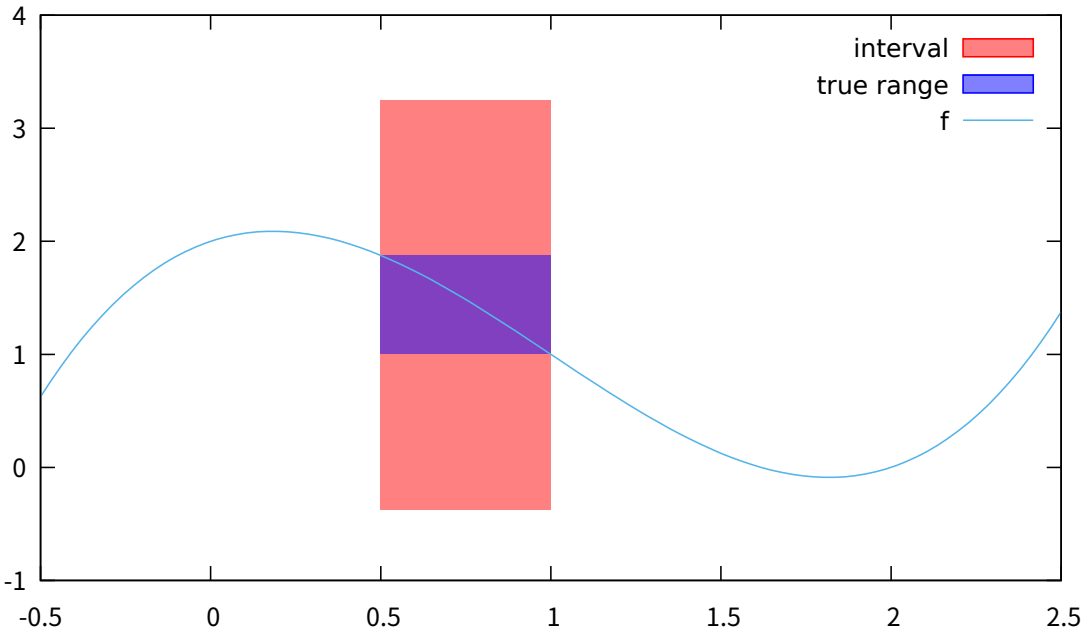
区間演算の実装上の注意点

- (入力誤差) プログラム中の定数 (`double x = 0.1;` の「`0.1`」など) はコンパイル時に 10 進数 →2 進数変換が行われ、その丸めの向きは制御できない。
- (出力誤差) 同様に、`std::cout << x << std::endl;` などの表示のときの 2 進数 →10 進数変換の丸めの向きも制御できない。
- (コンパイラの最適化) 最適化によって演算の順序が変えられたり定数をコンパイル時に計算されてしまうなどなどして、-O3 などで最適化を強くかけると丸めの向きの変更が効かないことがある。$\Longrightarrow$ `volatile` を使うなどして適切に最適化を抑制する。
- (数学関数) 数学関数で精度が保証されており丸めの向きの変更も出来るのは `sqrt` のみであり、`sqrt` 以外の関数は全く信用できない。

区間演算による関数の値域の評価

- 幅 0 の区間から出発して区間演算を行うと、丸め誤差の把握ができる。
- はじめから幅をもった区間から出発して区間演算を行えば、関数の値域の評価ができる。
- 区間演算の結果 $f(I)$ は、$f(I) \supset \{ f(x) \mid x \in I \}$ を満たす。

例: 関数 $f(x) = x^3 - 3x^2 + x + 2$ に区間 $I = [0.5, 1]$ を入れて区間演算すると、区間 $[-0.375, 3.25]$ を得る。真の像は、$[1, 1.875]$。

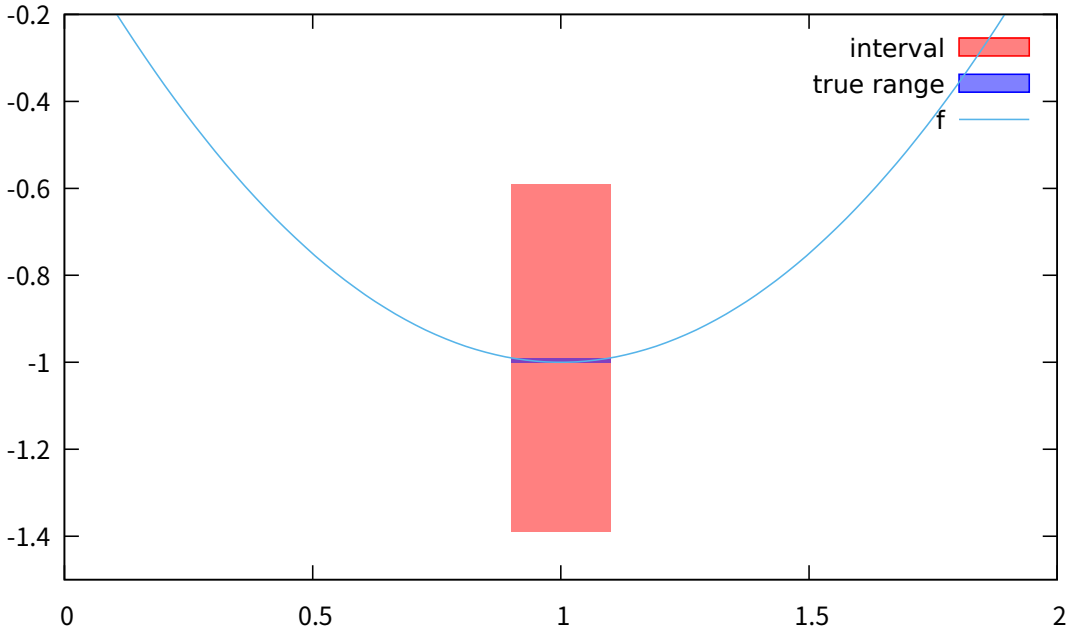


区間演算の過大評価

区間演算は、確かに真の値を含むものの、想像以上に区間幅が広がることがある。

$$f(x) = x^2 - 2x, \quad x \in [0.9, 1.1]$$

区間演算の結果は $[-1.39, -0.59]$ となるが、真の像は $[-1, -0.99]$。



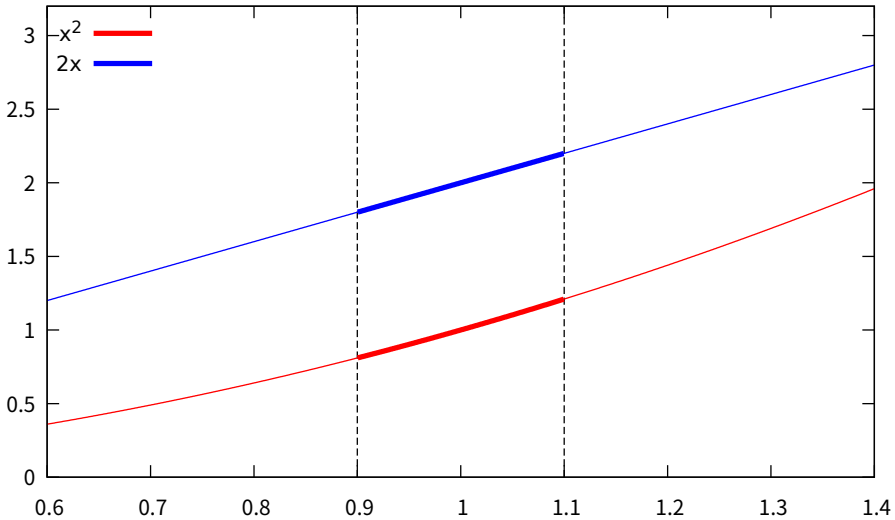
- 真の区間幅 0.01 に対して、区間演算の区間幅 0.8 (80 倍!)。
- この区間幅の拡大に、方向付き丸めによる拡大はまったく関係ない。

区間幅拡大の原因

$I = [0.9, 1.1]$ に対して、

- $I^2 = [0.81, 1.21]$ は、特に拡大していない。
- $2I = [1.8, 2.2]$ は、特に拡大していない。
- $I^2 - 2I = [-1.39, -0.59]$ は、極端に拡大している。ここの減算に問題がある。

区間演算が、変数間の相関性を取り扱えない (無視してしまう) ことによる。$[0.9, 1.1]$ で、関数 x^2 と $2x$ はほぼ同じ傾き 2 を持ち、片方が大きければもう片方も大きいというような相関性を持つが、区間演算の減算ではそんなことは分からないので、区間幅が増大してしまう。



- これが変数間の相関性の問題であることは、次の事実からも分かる。仮に、「$f(x, y) = x^2 - 2y, \quad x, y \in [0.9, 1.1]$」の像を求める問題であったならば、$[-1.39, -0.59]$ が真の像。

計算手順で結果が変わる

$x \in [0.9, 1.1]$ に対して:

- $f(x) = x^2 - 2x$ $[-1.39, -0.59]$ (区間幅 0.8)
- $f(x) = x(x-2)$ $[-1.21, -0.81]$ (区間幅 0.4)
- $f(x) = (x-1)^2 - 1$ $[-1, -0.99]$ (区間幅 0.01)

- 一般に数式中に同じ変数 (x) が 2 箇所以上出てくるものは苦手。
- よって、2 次式なら平方完成するのが最善。
- n 次式を区間演算するときどのような形がベストかは分からない。
  が、Horner 法の形は平均的に成績がよい。

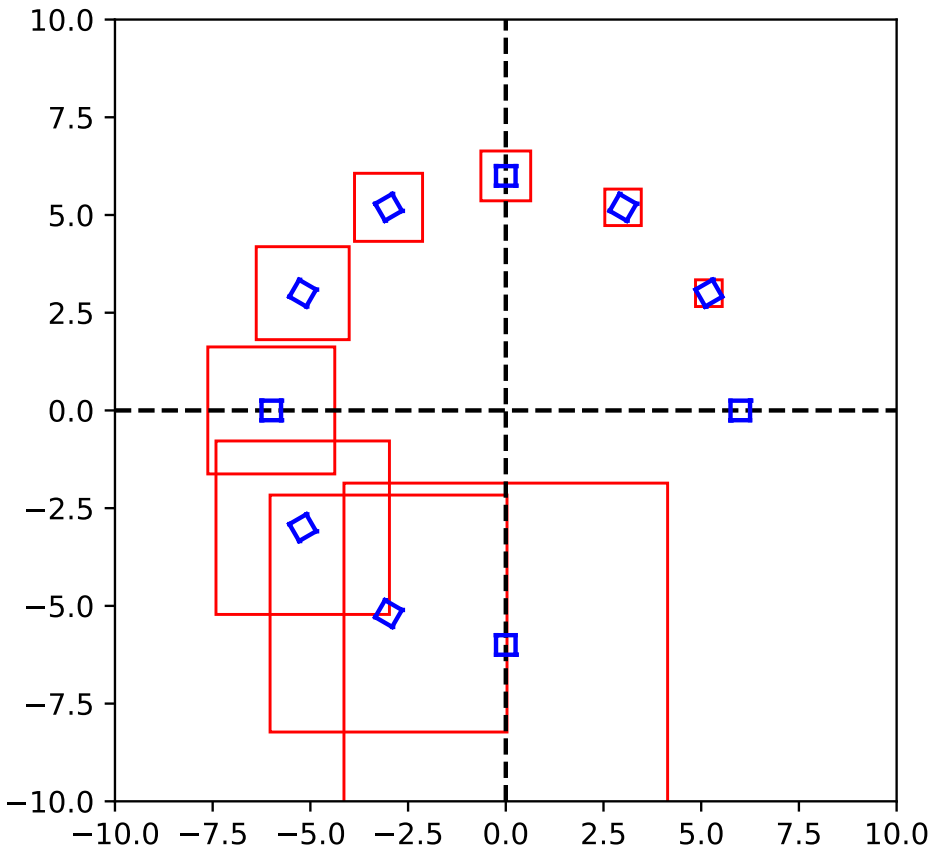
分配法則は成り立たない

$$X(Y+Z) \subset XY + XZ$$

例:

$$[-1,1](1+(-1)) = [0,0] \subset [-1,1] \times 1 + [-1,1] \times (-1) = [-2,2]$$

Wrapping Effect



成分が区間であるような「区間ベクトル」が $\mathbb{R}^n$ の辺が座標軸に平行な超直方体領域しか表現できないため、真の像がそのような直方体で包含されることにより区間幅が増大する現象。

区間 $I_0 = \begin{pmatrix} [5.75, 6.25] \\ [-0.25, 0.25] \end{pmatrix}$ に対して $I_{k+1} = \begin{pmatrix} \sqrt{3}/2 & -0.5 \\ 0.5 & \sqrt{3}/2 \end{pmatrix} I_k$ のように行列を作用させた例。

青が真の像、赤が区間演算の結果。

区間幅の増大を抑制する技術

区間演算の結果は確かに真の値を含むものの、ここまで見てきたように、さまざまな要因によって区間の幅が簡単に増大してしまい、それを抑制する技術が求められている。

平均値形式

- 関数の微分に対して区間演算を行い、間接的に値域を評価する。
- Krawczyk 法の基礎となっている。
- 自動微分 (の区間演算版) があれば容易に実装できる。
- 値域の評価を大きく改善することが多い。丸め誤差の把握の改善には役に立たない。

Affine Arithmetic

- 中間変数間の相関性を保持することによって区間幅の増大を抑制する。
- 計算量が大きい。
- 値域の評価、丸め誤差の把握の両方に効果がある。

平均値形式 (1)

平均値形式

$f(I)$ を直接評価するより、$c = \mathrm{mid}(I)$ として

$$f(c) + f'(I)(I - c)$$

を計算した方がよい場合が多い。

例: $f(x) = x^2 - 2x, \quad x \in [0.9, 1.1]$

直接評価

$$f(I) = [0.9, 1.1]^2 - 2[0.9, 1.1] = [0.81, 1.21] - [1.8, 2.2] = [-1.39, -0.59]$$

平均値形式

$f'(x) = 2x - 2$ なので、$f'(I) = 2[0.9, 1.1] - 2 = [-0.2, 0.2]$、
$c = \mathrm{mid}([0.9, 1.1]) = 1$、$f(c) = 1^2 - 2 \times 1 = -1$ なので、

$$f(c) + f'(I)(I - c) = -1 + [-0.2, 0.2]([0.9, 1.1] - 1) = [-1.02, -0.98]$$

平均値形式 (2)

関数 $f : \mathbb{R}^n \to \mathbb{R}^m$ に対して、区間ベクトル $I \subset \mathbb{R}^n$ の像を評価することを考える。$x, c \in I$ に対して積分型の平均値の定理

$$f(x) - f(c) = \int_0^1 f'(c + t(x - c))dt(x - c)$$

が成立する。ここで、

$$\int_0^1 f'(c + t(x - c))dt \in \mathrm{co}\left\{ f'(x) \,\middle|\, x \in I \right\} \subset f'(I)$$

(co は凸包、$f'(I)$ は $f'(x)$ を区間演算したもの) なので、$\forall x \in I$ について、

$$f(x) \in f(c) + f'(I)(x - c)$$

が成立する。また、$\forall x \in I$ について成り立つので x に I を代入して、

$$\{ f(x) \mid x \in I \} \subset f(c) + f'(I)(I - c)$$

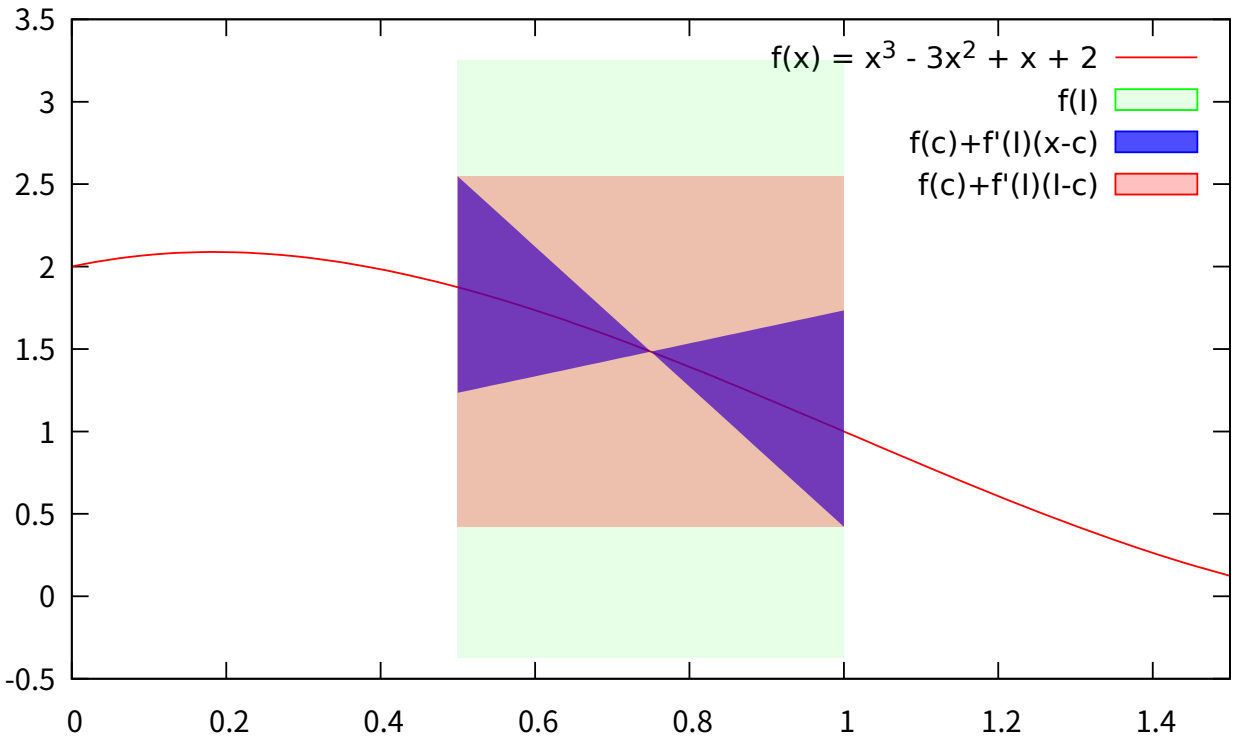
が成立する。(後者を平均値形式と呼ぶことが多いが、前者も重要)

平均値形式 (3)

関数 $f(x) = x^3 - 3x^2 + x + 2$、入力区間 $I = [0.5, 1]$ に対して、

- 区間演算
- 平均値形式

を適用した様子。緑が区間演算、青, 赤が平均値形式。



Taylor 形式

平均値形式を更に高次に拡張することもできる。Taylor 展開

$$f(x) = \sum_{k=0}^{n} \frac{1}{k!} f^{(k)}(c)(x-c)^k + R_{n+1}$$

に対して、Bernoulli の剰余項

$$R_{n+1} = \frac{1}{n!} \int_0^1 (1-t)^n f^{(n+1)}(c + t(x-c))dt(x-c)^{n+1}$$

を考える。ここで、 $s = (1-t)^{n+1}$ とおくと、

$$R_{n+1} = \frac{1}{(n+1)!} \int_0^1 f^{(n+1)}(c + (1 - \sqrt[n+1]{s})(x-c))ds(x-c)^{n+1}$$

と変型できるので、平均値形式と同様に考えて、

$$f(x) \in \sum_{k=0}^{n} \frac{1}{k!} f^{(k)}(c)(x-c)^k + \frac{1}{(n+1)!} f^{(n+1)}(I)(x-c)^{n+1}$$

が得られる。(平均値形式は $n = 0$ の場合に相当する。)

まとめ

区間演算

- 精度保証付き数値計算の基本技術。
- 数値を、計算機で表現可能な浮動小数点数を両端に持つ閉区間 $X = [a, b]$ で表現する。
- 区間同士の演算は、集合値演算として計算結果として有り得る値を包含するように行う。
- IEEE754 標準で定義されている「方向付き丸め」を利用する。
- 「丸め誤差の把握」と「関数の値域の評価」の 2 つの役割がある。

区間演算の区間幅の増大

- 区間演算は、確かに真の値を含むものの、想像以上に区間幅が広がることがある。
- 平均値形式を用いて区間幅の増大を抑制することができる。
- Affine Arithmetic については別の資料で説明する。