区分的線形近似の誤差推定
Error Estimation of Piecewise Linear Approximation

情報工学専攻 宮城 亮
Ryo MIYAGI

概要: 区分的微分可能なモデルに対する最適化は, 微分不可能点を含んでおり困難である. 本研究では Griewank らにより提唱されている Abs-Normal Form (ANF) を利用して一般化勾配を導出し, 区分線形化で関数を近似する際の誤差推定について考える. 区分的微分可能なモデルの中でも, 絶対値を用いたアルゴリズム (手続き) により計算される関数に対して, その近似値に含まれる誤差の推定方法について検討する. 結果として, 本研究の近似計算方法を用いて関数値の存在範囲を誤差領域から制限しうることが分かった.
キーワード: 区分的微分可能, 区分的線形近似, アルゴリズム微分

1 はじめに

近年, アルゴリズム微分において Griewank らにより提唱されている ANF による区分的線形近似を用いた微分不可能点での微分計算に進展があり, これにより関数 (アルゴリズム) の微分不可能点周辺の振る舞いが近似値から推定できるようになった. 本研究では, この近似値と関数値との誤差を推定する方法について考察する. ANF による区分的線形近似の計算方法を一部変更して導出した新たな近似値を用いて, 関数値の存在範囲を制限することを目的として, 数値実験を行い検証した. また, 数値実験のためアルゴリズム微分の実装をし, その計算コストについても実験を行いこの誤差推定を応用する際の指標として調査した.

2 アルゴリズム微分 [1]

アルゴリズム微分 (Algorithmic Differentation) とは, ある関数値を計算するアルゴリズム (プログラム) が与えられたとき, その偏導関数値を計算するアルゴリズム (プログラム) を導出する手法である. 数値微分や数式微分とは区別される.

2.1 基本演算

プログラム中で使用できる演算のことを基本演算 (Basic Operation) といい, 本研究では以下の演算 (関数) を対象とする.

- 四則演算 ($+, -, \times, \div$)
- 開平 (sqrt)
- 初等超越関数 (log, exp, sin , ...)
- 絶対値演算 (abs)

また, 本研究ではアルゴリズム微分の対象として, 最大値, 最小値関数 (max, min) を含んだ関数も想定している. それらの関数は abs により次のように定義する :

$$\max(u,w) = \frac{(u+w+\mathrm{abs}(u-w))}{2},$$
$$\min(u,w) = \frac{(u+w-\mathrm{abs}(u-w))}{2}.$$

2.2 中間変数と計算過程

基本演算の実行と一対一に対応し, その結果を格納する変数を中間変数 (Intermediate Variables) といい, 特に, アルゴリズムの入力値を格納する変数を入力変数 (Input Variables) , 最終的に計算された値を格納する中間変数を出力変数 (Output Variables) という. また, 1 つの基本演算の実行及び, その結果の中間変数への代入を計算ステップ (Computational Steps) という. 計算ステップの列, すなわち中間変数を逐次計算する過程を計算過程 (Computional Process) という. $\boldsymbol{y} = f(\boldsymbol{x})$ $(f : \mathbb{R}^n \to \mathbb{R}^m)$ を計算する手続きを次の計算過程で表現する [2] (式 (1)) :

$$\begin{aligned} u_{i-n} &= x_i, & i &= 1,...,n, \\ u_i &= \varphi_i(u_j)_{j\prec i}, & i &= 1,...,\ell, \\ y_i &= u_{\ell-m+i}, & i &= 1,...,m. \end{aligned} \tag{1}$$

φ_i は基本演算, u_i は中間変数である. 中間変数には, その計算順序に半順序関係がある.

2.3 実現方法 [3]

アルゴリズム微分の実現方法は大きく分けて, プログラミング言語の機能であるオペレーターオーバーロードを用いて演算子に微分係数を計算するアルゴリズムを追加定義することで実現する Operator Overloading 型と, プログラムを読み込んでアルゴリズム微分を実行するプログラムを新たに生成するプリプロセッサ, プリコンパイラによって実現する Source Code Translation 型の 2 つがある.

本研究では C++ のオペレーターオーバーロードを用いて Operator Overloading 型を実装した.

3 区分的微分可能 [3]

3.1 区分的微分可能関数

開集合 $\mathcal{D} \subset \mathbb{R}^n$ が与えられたとき, $\boldsymbol{f} : \mathcal{D} \to \mathbb{R}^m$ が $\mathcal{N}$ 上で連続する開近傍 $\mathcal{N} \subset \mathcal{D}$ と, C^1 級関数 $f_i : \mathcal{N} \to \mathbb{R}^m (i = 1,...,k)$ が存在するとする. 関数 f が以下を満たすとき, 関数 f は点 $\boldsymbol{x} \in \mathcal{D}$ で区分的微分可能であるという.

$$\boldsymbol{f}(\boldsymbol{x}) \in f_1(\boldsymbol{x}),...,f_k(\boldsymbol{x}) \quad \text{for } \boldsymbol{x} \in \mathcal{N} \tag{2}$$

このとき, $f_i(\boldsymbol{x})$ がすべて線形であるならば, $\boldsymbol{f}$ は区分的線形関数という. また, 関数 $\boldsymbol{f}$ における微分不可能点を kink point という.

3.2 局所リプシッツ連続

関数 $f : \mathbb{R}^n \to \mathbb{R}$ が点 $\boldsymbol{x} \in \mathbb{R}^n$ で局所リプシッツ連続 (Locally Lipschitz Continuous) であるとは, ある $K > 0, \epsilon > 0 (K, \epsilon \in \mathbb{R})$ が存在して, $\boldsymbol{x}$ の ϵ 近傍 $B(\boldsymbol{x};\epsilon)$ に対して, 以下が成り立つことをいう (4).

$$|f(\boldsymbol{x''}) - f(\boldsymbol{x'})| \le K|\boldsymbol{x''} - \boldsymbol{x'}| \quad \text{for all } \boldsymbol{x''}, \boldsymbol{x'} \in B(\boldsymbol{x};\epsilon) \tag{3}$$

関数 $f : \mathbb{R}^n \to \mathbb{R}$ が $\mathcal{D} \in \mathbb{R}^n$ 上で局所リプシッツ連続であるとは, $\mathcal{D}$ に属するすべての点で以下が成り立つことをいう (4).

$$|f(\boldsymbol{x''}) - f(\boldsymbol{x'})| \le K|\boldsymbol{x''} - \boldsymbol{x'}| \quad \text{for all } \boldsymbol{x''}, \boldsymbol{x'} \in \mathcal{D} \tag{4}$$

区分的微分可能関数は局所リプシッツ連続であることが知られている.

3.3 区分的微分可能レベル [4]

区分的微分可能関数を構成要素からいくつかのレベルに分類することにする.

レベル 0 (Smooth)
定義域内に微分不可能点が存在しない滑らかな関数.

レベル 1 (Piecewise Smooth)
局所リプシッツ連続であるが, 定義域内に微分不可能点を含む関数. その微分不可能点は abs, min, max によって表現される.

レベル 2 (Piecewise Semismooth)
局所リプシッツ連続であるが, 定義域内に微分不可能点を含む関数. その微分不可能点はレベル 1 にユークリッドノルムを加えたものによって表現される.

レベル 3 (Discontinuous Piecewise Smooth)
不連続な関数. その微分不可能点はレベル 2 に sign 関数と条件分岐を加えたものによって表現される.

このレベルが高いほど, 微分不可能点の周辺の扱いが困難になる.

本研究では, アルゴリズム微分の対象となる関数 (アルゴリズム) はすべてレベル 1 とする.

4 Abs-Normal Form [5, 6, 7]

絶対値を含む式 $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})(\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m)$ に対して, 絶対値演算を境にわけて表現する. このような計算過程の表現を ANF という (式 (5)) :

$$\begin{aligned} \boldsymbol{w} &= \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{z}), \\ \boldsymbol{z} &= |\boldsymbol{w}|, \\ \boldsymbol{y} &= \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{z}), \end{aligned} \tag{5}$$

ただし, 各変数は $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{w} \in \mathbb{R}^s$, $\boldsymbol{z} \in \mathbb{R}^s$ であり, $\boldsymbol{g} : \mathbb{R}^{n+s} \to \mathbb{R}^s$, $\boldsymbol{h} : \mathbb{R}^{n+s} \to \mathbb{R}^m$, は連続微分可能関数, $\boldsymbol{z} = |\boldsymbol{w}|$ は成分ごとの絶対値を表す.

4.1 基本演算の増分

各基本演算の増分の近似を以下のように考える (式 (6)) :

$$\begin{aligned} \Delta u_i &= \Delta u_j \pm \Delta u_k && \text{when} \quad u_i = u_j \pm u_k, \\ \Delta u_i &= u_j \Delta u_k + u_k \Delta u_j && \text{when} \quad u_i = u_j \cdot u_k, \\ \Delta u_i &= \varphi_i'(u_j) \Delta u_j && \text{when} \quad u_i = \varphi_i(u_j), \\ \Delta u_i &= |u_j + \Delta u_j| - u_i && \text{when} \quad u_i = |u_j|, \end{aligned} \tag{6}$$

最終的に近似されている増分 $\Delta \boldsymbol{y_i}$ を $\Delta f(\boldsymbol{x}; \Delta \boldsymbol{x})$ と記す. φ_i は連続微分可能な関数である.

このとき, 関数 f の点 $\boldsymbol{x}$ での $\Delta \boldsymbol{x}$ 方向の区分線形近似 $f_{PL,\boldsymbol{x}}(\Delta \boldsymbol{x})$ は以下のように表現できる (式 (7)) :

$$f_{PL,\boldsymbol{x}}(\Delta \boldsymbol{x}) = f(\boldsymbol{x}) + \Delta f(\boldsymbol{x}; \Delta \boldsymbol{x}). \tag{7}$$

4.2 ANF の区分線形化

ANF (式 (5)) に基づいて点 $\mathring{\boldsymbol{x}}$ からの $\Delta \boldsymbol{x}$ 方向の増分は, 各変数の 点 $\mathring{\boldsymbol{x}}$ での計算式

$$\begin{aligned} \mathring{\boldsymbol{w}} &= \boldsymbol{g}(\mathring{\boldsymbol{x}}, \mathring{\boldsymbol{z}}), \\ \mathring{\boldsymbol{z}} &= |\mathring{\boldsymbol{w}}|, \\ \mathring{\boldsymbol{y}} &= \boldsymbol{h}(\mathring{\boldsymbol{x}}, \mathring{\boldsymbol{z}}) \end{aligned} \tag{8}$$

を用いて,

$$\begin{gathered} \boldsymbol{g}(\mathring{\boldsymbol{x}} + \Delta \boldsymbol{x}, \mathring{\boldsymbol{z}} + \Delta \boldsymbol{z}) - \mathring{\boldsymbol{w}}, \\ |\mathring{\boldsymbol{w}} + \Delta \boldsymbol{w}| - \mathring{\boldsymbol{z}}, \\ \boldsymbol{h}(\mathring{\boldsymbol{x}} + \Delta \boldsymbol{x}, \mathring{\boldsymbol{z}} + \Delta \boldsymbol{z}) - \mathring{\boldsymbol{y}} \end{gathered} \tag{9}$$

となる.

これらから, $\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{y}$ の増分の区分的線形近似を以下のように表現する (式 (10)) :

$$\begin{aligned} \Delta \boldsymbol{w} &= \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}(\mathring{\boldsymbol{x}}, \mathring{\boldsymbol{z}}) \Delta \boldsymbol{x} + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}}(\mathring{\boldsymbol{x}}, \mathring{\boldsymbol{z}}) \Delta \boldsymbol{z}, \\ \Delta \boldsymbol{z} &= |\mathring{\boldsymbol{w}} + \Delta \boldsymbol{w}| - \mathring{\boldsymbol{z}}, \\ \Delta \boldsymbol{y} &= \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}(\mathring{\boldsymbol{x}}, \mathring{\boldsymbol{z}}) \Delta \boldsymbol{x} + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}(\mathring{\boldsymbol{x}}, \mathring{\boldsymbol{z}}) \Delta \boldsymbol{z}. \end{aligned} \tag{10}$$

ここで,

$$\begin{aligned} \widetilde{\Delta \boldsymbol{w}} &= \Delta \boldsymbol{w} + \mathring{\boldsymbol{w}}, \\ \widetilde{\Delta \boldsymbol{z}} &= \Delta \boldsymbol{z} + \mathring{\boldsymbol{z}} = |\widetilde{\Delta \boldsymbol{w}}| \end{aligned} \tag{11}$$

を導入すると $\Delta \boldsymbol{y}$ は以下のように表現できる (式 (12)) :

$$\begin{aligned} \widetilde{\Delta \boldsymbol{w}} &= \mathring{\boldsymbol{w}} + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}} \Delta \boldsymbol{x} + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}}(\widetilde{\Delta \boldsymbol{z}} - \mathring{\boldsymbol{z}}), \\ \widetilde{\Delta \boldsymbol{z}} &= \Sigma \widetilde{\Delta \boldsymbol{w}}, \\ \Delta \boldsymbol{y} &= \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \Delta \boldsymbol{x} + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}}(\widetilde{\Delta \boldsymbol{z}} - \mathring{\boldsymbol{z}}). \end{aligned} \tag{12}$$

Σ は各絶対値演算の引数の符号を格納する行列である. 具体的な定義を以下に示す. 符号関数 (式 (13)) を用いて,

$$\mathrm{sign(a)} = \begin{cases} 1 & (a > 0) \\ 0 & (a = 0) \\ -1 & (a < 0) \end{cases} \tag{13}$$

絶対値演算は $|w| = \mathrm{sign}(w) \cdot w$ と書き換えられるので, 次のような対角行列を Σ として定義する :

$$\Sigma = \begin{pmatrix} \mathrm{sign}(w_1 + \Delta w_1) & & \\ & \ddots & \\ & & \mathrm{sign}(w_s + \Delta w_s) \end{pmatrix}. \tag{14}$$

式 (12) を定数項をまとめて

$$\begin{aligned} \boldsymbol{w_c} &= \mathring{\boldsymbol{w}} - \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}} \mathring{\boldsymbol{z}}, \\ \boldsymbol{y_c} &= -\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \mathring{\boldsymbol{z}} \end{aligned} \tag{15}$$

行列で表現すると以下のようになる (式 (16)) :

$$\begin{aligned} \begin{pmatrix} \widetilde{\Delta \boldsymbol{w}} \\ \Delta \boldsymbol{y} \end{pmatrix} &= \begin{pmatrix} \boldsymbol{w_c} \\ \boldsymbol{y_c} \end{pmatrix} + \begin{bmatrix} \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}} \\ \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} & \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x} \\ \widetilde{\Delta \boldsymbol{z}} \end{bmatrix}, \\ \widetilde{\Delta \boldsymbol{z}} &= \Sigma \widetilde{\Delta \boldsymbol{w}}. \end{aligned} \tag{16}$$

これを $\Delta \boldsymbol{x}$ について解くと, $\boldsymbol{y} = \mathring{\boldsymbol{y}}$ における区分的線形近似 $\mathring{\boldsymbol{y}} + \Delta \boldsymbol{y}$ を得る (式 (17)) :

$$\begin{aligned} \mathring{\boldsymbol{y}} + \Delta \boldsymbol{y} &= \boldsymbol{c} + (\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \Sigma \boldsymbol{A} \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}) \Delta \boldsymbol{x}, \\ \boldsymbol{A} &= (\boldsymbol{I} - \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}} \Sigma)^{-1}, \\ \boldsymbol{c} &= \mathring{\boldsymbol{y}} + \boldsymbol{y_c} + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \Sigma \boldsymbol{A} \boldsymbol{w_c}. \end{aligned} \tag{17}$$

5 近似値の誤差推定

第 4 節で述べた手法をもとに, 区分的微分可能な関数に対する微分計算と線形近似を行う. 本章では, この計算による関数 (アルゴリズム) の kink point 周辺における近似値と実測値の誤差の推定方法について考察する.

5.1 ANF による区分的線形近似の誤差

ANF による区分的線形近似の特徴は, 目的関数の kink point (微分不可能点), すなわち計算過程に現れるいずれか, あるいはすべての絶対値演算の引数が 0 になる点と, その絶対値演算の符号が切り替わった先の情報を推測 (近似) できることである.

しかし, 近似である以上計算により求めた kink point の座標も関数値も実測値とは異なり, また, 計算機上で正確な 0 の値を制御することは難しく, プログラムにより計算される ANF を用いた kink point の近似値は様々な誤差を含んでいる.

本研究では, 数値計算上の誤差でなく近似値としての実測値との誤差について考察する.

5.2 kink point 周辺の近似

ANF によって区分的微分可能な関数の微分不可能点での微分計算が可能であることから, 本研究では, ANF による区分的線形近似から目的関数 (アルゴリズム) の kink point を探索することを想定する.

基準点 $\mathring{\boldsymbol{x}}$ から, ある kink point $\boldsymbol{x}_{kink}$ に対し, $\mathring{\boldsymbol{x}} + \Delta\boldsymbol{x}_{tokink} = \boldsymbol{x}_{kink}$ となるような $\Delta\boldsymbol{x}_{tokink}$ について考える. $\boldsymbol{x}_{kink}$ では, いずれかの絶対値演算の引数が 0 となっているため, $\exists k \;\; \mathring{w}_k + \Delta w = 0$ すなわち,

$$\exists k \;\; \widetilde{\Delta w_k} = 0 \tag{18}$$

であるので, ANF による区分的線形近似の計算式 (式 (11)) から,

$$\begin{aligned} \widetilde{\Delta \boldsymbol{w}} &= \boldsymbol{w}_c + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\Delta\boldsymbol{x} + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}}\widetilde{\Delta \boldsymbol{z}} \\ &= \boldsymbol{w}_c + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\Delta\boldsymbol{x} + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}}\Sigma\widetilde{\Delta \boldsymbol{w}} \end{aligned} \tag{19}$$

$\widetilde{\Delta \boldsymbol{w}}$ について解くと,

$$\widetilde{\Delta \boldsymbol{w}} = (\boldsymbol{I} - \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}}\Sigma)^{-1} \cdot (\boldsymbol{w}_c + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\Delta\boldsymbol{x}) \tag{20}$$

となる. 式 (18) より, 引数を 0 とする絶対値演算に対して, 式 (20) の対応する $\widetilde{\Delta w_k}$ に 0 を代入して $\Delta\boldsymbol{x}$ について解けばよい. 導出した $\boldsymbol{x}_{kink}$ への $\boldsymbol{x}$ の増分 $\Delta\boldsymbol{x}_{tokink}$ をもとに, そこから更に値をずらすことで, kink point 周辺の関数値の近似が求まる.

5.3 絶対値演算の増分の近似

本研究では, kink point 周辺の近似値の誤差に関して絶対値演算の増分の近似に着目した. ANF による絶対値演算の増分 $\Delta\boldsymbol{z}$ は式 (9) より, 以下のように表現できる (式 (21)).

$$\begin{aligned} \Delta\boldsymbol{z} = \;& |\mathring{\boldsymbol{w}} + \Delta\boldsymbol{w}| - \mathring{\boldsymbol{z}} \\ =\; & |\mathring{\boldsymbol{w}} + \Delta\boldsymbol{w}| - |\mathring{\boldsymbol{w}}| \\ =\; & \mathrm{sign}(\mathring{\boldsymbol{w}} + \Delta\boldsymbol{w}) \cdot (\mathring{\boldsymbol{w}} + \Delta\boldsymbol{w}) - \mathrm{sign}(\mathring{\boldsymbol{w}}) \cdot \mathring{\boldsymbol{w}} \\ =\; & \mathrm{sign}(\mathring{\boldsymbol{w}} + \Delta\boldsymbol{w}) \cdot \Delta\boldsymbol{w} + (\mathrm{sign}(\mathring{\boldsymbol{w}} + \Delta\boldsymbol{w}) - \mathrm{sign}(\mathring{\boldsymbol{w}})) \cdot \mathring{\boldsymbol{w}} \\ =\; & \Sigma\Delta\boldsymbol{w} + \Sigma_0\mathring{\boldsymbol{w}} \end{aligned} \tag{21}$$

Σ_0 は Σ と同様に対角成分に絶対値演算の符号に関する値を格納した行列である. 詳細な定義を以下に示す (式 (22)).

$$\Sigma_0 = \begin{pmatrix} \mathrm{sign}(\mathring{w}_1 + \Delta w_1) - \mathrm{sign}(\mathring{w}_1) & & \\ & \ddots & \\ & & \mathrm{sign}(\mathring{w}_s + \Delta w_s) - \mathrm{sign}(\mathring{w}_s) \end{pmatrix} \tag{22}$$

ここで, 絶対値演算の引数 $\mathring{w}_k + \Delta w_k$ の値を 0 と判定した添え字 k の集合を K とする ($k \in K$). 基準点 $\mathring{\boldsymbol{x}}$ から初めて K が空集合でなくなる点 $\mathring{\boldsymbol{x}} + \Delta\boldsymbol{x}_{tokink}$ では, 行列 Σ の対角成分 (i,i) を σ_i と表現すると

$$\sigma_k = 0 \text{ for all } k \in K \tag{23}$$

となる.
一方, 行列 Σ_0 について基準点 $\mathring{\boldsymbol{x}}$ では $\Delta\boldsymbol{x} = \boldsymbol{0}$ より $\Delta\boldsymbol{w} = \boldsymbol{0}$ なので,

$$\mathrm{sign}(\mathring{\boldsymbol{w}} + \Delta\boldsymbol{w}) = \mathrm{sign}(\mathring{\boldsymbol{w}}) \tag{24}$$

となり, 行列 Σ_0 の定義から零行列となる.
また, 点 $\mathring{\boldsymbol{x}} + \Delta\boldsymbol{x}_{tokink}$ でも $\mathring{w}_k + \Delta w_k = 0$ ($k \in K$) より, 同様に行列 Σ_0 の対角成分 (i,i) を $(\sigma_0)_i$ と表現すると

$$(\sigma_0)_k = 0 \text{ for all } k \in K \tag{25}$$

となる. よって, (k,k) 成分を含むすべての成分が 0 となるので零行列となる.
しかし, この点から更に $\Delta\boldsymbol{x}_{tokink}$ 方向に移動した点 $\mathring{\boldsymbol{x}} + (1+\delta)\Delta\boldsymbol{x}_{tokink}$ $(\delta > 0)$ では, $\mathrm{sign}(\mathring{w}_k + (1+\delta)\Delta w_k) = -\mathrm{sign}(\mathring{w}_k)$ より,

$$(\sigma_0)_k = \begin{cases} -2 & (\mathring{w}_k > 0) \\ 2 & (\mathring{w}_k < 0) \end{cases} \text{ for all } k \in K \tag{26}$$

となり, 行列 Σ_0 は零行列ではなくなる. 基準点 $\mathring{\boldsymbol{x}}$ から段階的に $\Delta\boldsymbol{x}$ を更新する場合, ANF による区分的線形近似上で $\boldsymbol{x}_{kink}$ を探索するには, Σ の値に注目してある成分が 0 になるまで, あるいは $\boldsymbol{x}_{kink}$ に至る前の各対角成分の値を保持して, その値が切り替わるまで $\Delta\boldsymbol{x}$ を更新するよりも, Σ_0 の値に注目しての成分に 0 以外の値が現れるまで $\Delta\boldsymbol{x}$ を更新する方が作業用変数が少なく簡潔に実装できる.

5.4 絶対値演算の符号の変更による近似値の変化

ANF による区分的線形化における kink point 周辺の近似値の変化は, 絶対値演算の増分の近似の符号の変化, すなわち Σ, Σ_0 の値の変化によって発生する.

本研究では, この区分的線形近似によって発生する誤差について, kink point の周辺では絶対演算の近似を変更することで関数値との誤差を抑えることができるのか, または関数値の存在範囲を限定する要素となり得るのかを検証した.

そこで, kink point の周辺の点 $\mathring{\boldsymbol{x}} + (1+\delta)\Delta\boldsymbol{x}_{tokink}$ において, kink point で絶対値演算の計算値が 0 になったと推測できるものに対して, 対応する Σ , Σ_0 またはその両方の値を意図的に変更して区分的線形近似を計算する新たな近似値について考察する.

ここで, 説明のため記号を導入する. また, 各 Σ , Σ_0 の具体的な値の変更についてはそれぞれ第 5.4.1 項, 第 5.4.2 項, 第 5.4.3 項にて述べる.

1 つ以上の絶対値演算の引数 $w_k + \Delta w_k$ が 0 と思われるものに対して, Σ , Σ_0 , またはその両方の第 (k,k) 成分を意図的に変更して計算した関数 (アルゴリズム) の増分を以下のように表現する (式 (27)).

$$\Delta\bar{f}_{\boldsymbol{p}}(\mathring{\boldsymbol{x}}; \Delta\boldsymbol{x}) \tag{27}$$

ベクトル $\boldsymbol{p}$ は 0 または 1 の値から構成される, 変更した Σ , Σ_0 またはその両方の成分番号を表す変数である. 例えば, 絶対値演算が 5 つあり, そのうちの 2 番目 と 5 番目を意図的に 0 に変更した場合の ベクトル $\boldsymbol{p}$ は

$$\boldsymbol{p} = (0\ 1\ 0\ 0\ 1)^{\mathrm{T}}$$

と表現する.

Σ_0 の値を変更して計算した近似値は以下のように表現する (式 (28)).

$$\bar{f}_{PL,\mathring{\boldsymbol{x}},\boldsymbol{p}}(\Delta\boldsymbol{x}) = f(\mathring{\boldsymbol{x}}) + \Delta\bar{f}_{\boldsymbol{p}}(\mathring{\boldsymbol{x}}; \Delta\boldsymbol{x}) \tag{28}$$

実際に kink point 周辺の点 $\mathring{\boldsymbol{x}} + (1+\delta)\Delta\boldsymbol{x}_{tokink}$ での Σ_0 の意図的な値の変更による近似値 $\bar{f}_{PL,\mathring{\boldsymbol{x}},\boldsymbol{p}}((1+\delta)\Delta\boldsymbol{x}_{tokink})$ の変化を検証する際は, 絶対値演算の引数 $\boldsymbol{w} + (1+\delta)\Delta\boldsymbol{w}_{tokink}$ について, 適当な閾値 μ $(0 < \mu)$ を用いて

$$|w_k + (1+\delta)\Delta w_{tokink_k}| < \mu \tag{29}$$

となるような k に対し, すべてのパターンの Σ , Σ_0 , またはその両方の第 (k,k) 成分を 0 とした計算をしなければならない. 値の変更パターンの総数は $2^{|K|} - 1$ 種類である.

5.4.1 Σ と Σ_0 の値の変更による近似値

kink point 周辺において絶対値演算の引数が 0 になったと推測できるものに対して, その絶対値演算の増分の符号の切り替えを完全に無視した近似値を考える. これは, 絶対値演算の引数が 0 に近い値で演算結果に符号の切り替えが発生した場合, その引数に至るまでの近似計算で発生した誤差によって符号が切り替わったと判断して, 符号の変化をなくすことで誤差を抑える意図がある.

具体的には, Σ の 第 (k,k) 成分の値を 1 に, Σ_0 の 第 (k,k) 成分は 0 に変更する. これにより変更して計算した絶対値演算の増分 $\overline{\Delta z_k}$ は,

$$\overline{\Delta z_k} = 1 \cdot \Delta w_k + 0 \cdot \mathring{w}_k = \Delta w_k \tag{30}$$

となり, k 番目の絶対値演算を無視した増分の近似を得る.

5.4.2 Σ の値の変更による近似値

第 5.4.1 項とは異なる絶対値の増分の近似計算として, ここでは Σ の 第 (k,k) 成分を 1 として絶対値演算の増分を計算する. これにより変更して計算した絶対値演算の増分 $\overline{\Delta z_k}$ は,

$$\overline{\Delta z_k} = 1 \cdot \Delta w_k + (\mathrm{sign}(\mathring{w}_k + \Delta w_k) - \mathrm{sign}(\mathring{w}_k)) \cdot \mathring{w}_k \quad (31)$$

となる.

5.4.3 Σ_0 の値の変更による近似値

最後に, Σ_0 の 第 (k,k) 成分を 0 として絶対値演算の増分を計算する. これにより変更して計算した絶対値演算の増分 $\overline{\Delta z_k}$ は,

$$\begin{aligned} \overline{\Delta z_k} &= \mathrm{sign}(\mathring{w}_k + \Delta w_k) \cdot \Delta w_k + 0 \cdot \mathring{w}_k \\ &= \mathrm{sign}(\mathring{w}_k + \Delta w_k) \cdot \Delta w_k \end{aligned} \quad (32)$$

となる.

5.4.4 数値実験

絶対値演算の増分の近似計算を変更して計算する近似値について, その 3 種類 (第 5.4.1 項, 第 5.4.2 項 , 第 5.4.3 項) による近似値とその点での計算により求めた関数値との誤差領域をまとめた表を以下に示す. ここではテスト関数として, 文献 [8] の第 9.1 項に掲載されている Small Unconstrained Problems (以下 SUP) の一部を使用した.

H

表 1　手法ごとの関数値との誤差領域

$f(\boldsymbol{x})$	$n(K)$	手法	誤差領域
SUP 1	1	手法 5.4.1	[-0.002420 , 0.000903]
		手法 5.4.2	[1.48e-06 , 0.000903]
		手法 5.4.3	[-0.002420 , 0.000903]
SUP 2	2	手法 5.4.1	[-0.003504 , 5.93e-06]
		手法 5.4.2	[5.93e-06 , 0.000404]
		手法 5.4.3	[-0.003504 , 5.93e-06]
SUP 3	2	手法 5.4.1	[-0.006505 , 4.44e-16]
		手法 5.4.2	[4.44e-16 , 0.002400]
		手法 5.4.3	[-0.006505 , 4.44e-16]
SUP 4	1	手法 5.4.1	[0.009132 , 0.012137]
		手法 5.4.2	[0.009132 , 0.075086]
		手法 5.4.3	[-0.062938 , 0.009132]
SUP 5	1	手法 5.4.1	[-0.464741 , 0.042470]
		手法 5.4.2	[0.042323 , 0.042470]
		手法 5.4.3	[-0.464741 , 0.042470]
SUP 6	1	手法 5.4.1	[0.003852 , 0.309235]
		手法 5.4.2	[0.003852 , 0.476833]
		手法 5.4.3	[-0.167598 , 0.003852]
SUP 21	1	手法 5.4.1	[0.000299 , 0.000484]
		手法 5.4.2	[0.000299 , 0.002530]
		手法 5.4.3	[-0.002046 , 0.000299]
SUP 22	1	手法 5.4.1	[0.000710 , 0.008867]
		手法 5.4.2	[0.000710 , 0.288207]
		手法 5.4.3	[-0.278949 , 0.000710]
SUP 23	2	手法 5.4.1	[0.286453 , 0.505526]
		手法 5.4.2	[0.286453 , 0.511465]
		手法 5.4.3	[-0.834818 , 0.109232]
SUP 38	1	手法 5.4.1	[0.000874 , 0.002869]
		手法 5.4.2	[0.000874 , 0.005618]
		手法 5.4.3	[-0.002747 , 0.000874]
SUP 39	2	手法 5.4.1	[-1.990000 , 0.020200]
		手法 5.4.2	[-1.990000 , 0.020200]
		手法 5.4.3	[-1.990000 , 4.44e-16]
SUP 8	2	手法 5.4.1	[0.000452 , 0.012034]
		手法 5.4.2	[0.000452 , 0.012034]
		手法 5.4.3	[-0.061908 , 0.000452]
SUP 40	3	手法 5.4.1	[-0.004839 , 4.09e-05]
		手法 5.4.2	[-0.007661 , 4.09e-05]
		手法 5.4.3	[-0.004839 , 4.09e-05]

この結果より使用したすべてのテスト関数に対して, 第 5.4.3 項による絶対値演算の増分の近似の変更により導出した近似値と, 通常の ANF による区分的線形近似の近似値を用いて関数値との誤差を計算すると, 必ず正と負の値が出現している. このことから, 関数値はこの近似値による誤差領域内に存在することが分かり, 近似値から関数値の存在範囲を制限しうることを確認できた.

6　結論

本研究では, 微分不可能点をアルゴリズム微分で扱うために ANF による区分的線形近似を利用し, 関数の微分不可能点周辺での近似値を求め, 実測値との誤差について絶対値の増分を変更して導出した新たな近似値について検証した.

具体的には, 微分不可能点周辺では絶対値演算の符号の切り替わりによって近似値が変わる. この符号の切り替わりを意図的に変更した近似値を計算し, 関数値との誤差を求めた.

実験から使用した区分的微分可能なテスト関数について, 通常の ANF による区分的線形近似から求めた近似値と Σ_0 に変更を加えて計算する区分的線形近似から求めた近似値を利用して, 関数値との誤差領域を求め, その存在範囲を限定し推定することができた.

7　今後の課題

誤差領域の上界と下界を得られるベクトル $\boldsymbol{p}$ がわからないため, 現状 $2^{|K|}-1$ 通りのすべての計算パターンを試行しなければならない. 実際に誤差領域として使用する近似値は上界と下界の 2 つしかなく, 無駄な近似値を計算する回数を減らす必要がある. また, 数値計算上の誤差について考慮する必要もある.

より多くの関数に対して近似値の比較を行い, 本研究で扱いきれなかったベクトル $\boldsymbol{p}$ や近似計算による誤差だけでなく数値計算上での誤差も含めた誤差について理解を深めていきたい.

参考文献

[1] 久保田 光一, 伊理 正夫, **“アルゴリズムの自動微分と応用”**, コロナ社, 1998.

[2] 吉田 北斗, 久保田 光一, **“区分的微分可能関数と一般化自動微分”**, 情報処理学会 第 79 回全国大会予稿集, vol. I, p. 243, 2017.

[3] 吉田 北斗, **“区分的微分可能関数と一般化自動微分”**, 中央大学理工学研究科情報工学専攻, 2017.

[4] A. Griewank, **“Treatement of nonsmooth problems via algorithmic piecewise differentiation,”** Presentation at Chuo University, October, 2015.

[5] A. Griewank, **“On stable piecewise linearization and generalized algorithmic differentaion,”** Optimization Methods and Software, vol. 28, pp. 1139–1178, Dec. 2013.

[6] A. Griewank, Jens-Uwe Bernt, Manuel Radons and Tom Streubel, **“Solving piecewise linear equations in abs-normal form”**, Linear Algebra and its Application, vol. 471, pp. 500–530, 2015.

[7] A. Griewank, A. Walther, S. Fiege, and T. Bosse, **“On Lipschitz optimization based on gray-box piecewise linearization,”** Mathematical Programming, vol. 158, no. 1, pp.383–415, 2016.

[8] A. Bagirov, N. Karmitsa, M. Mäkelä, **“Introduction to Nonsmooth Optimization,”** Springer, 2014.