整数計画法を用いた区分的線形区間方程式の全ての解集合を求める方法

Finding All Solution Sets of Piecewise-Linear Interval Equations Using Integer Programming

電気電子情報通信工学専攻　高根　裕一郎
Yuichiro　TAKANE

1. まえがき

非線形回路の設計・解析において，非線形素子の特性が種々の原因により変動を起こしたとき，回路全体の特性がどのように変動するかを見積もることは，より品質の高い，信頼性の高い回路を設計する上で重要となる．このような変動解析では，非線形抵抗の素子特性を “集合値写像” で表すことがしばしば有効となる．

文献 [1] では，非線形素子の特性が集合値写像となるような非線形回路を区分的線形区間方程式（PLI 方程式）で表す方法が示されている．

近年，整数計画法の分野の飛躍的な発展により，10 年前までは NP 困難という呪縛から「絶対に」解けないと考えられていた大規模な整数計画問題を実用的な計算時間で解けるようになっている．

このような整数計画法の飛躍的な発展に着目し，文献 [2] では混合整数計画問題を整数計画ソルバー CPLEX で 1 回解くだけで非線形回路の全ての解（直流動作点）が得られることを示した．この方法は複雑なプログラミングを必要としないため実装が容易で，かつ非常に効率が良い．

文献 [3] では，CPLEX を用いた PLI 方程式の全ての解集合を求める方法を提案した．この問題は「点」ではなく「連結されていない複数の非凸多面体の集合」を求める問題となるため，文献 [2] の単純な拡張ではできない，難しい問題となる．しかし，この方法は混合整数計画問題を $2n$ 回も解く必要があり，効率的ではなかった．

本論文では，文献 [3] の方法の計算効率を大幅に改善した，PLI 方程式の全ての解集合を求める新しい方法を提案する．本手法は混合整数計画問題を 1 回解くだけで全ての解集合を求めることができる．本手法も整数計画ソルバー CPLEX を用いるもので，インプリメントは非常に容易である．また，本手法によって全ての解集合が得られることを，CPLEX で使われているアルゴリズムの原著論文とマニュアルから証明する．

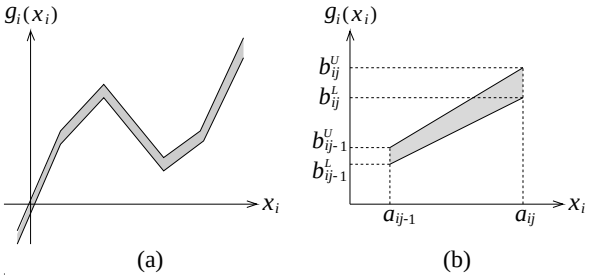

図 1　区分的線形区間関数

2. 従来法

本章ではまず，提案手法のベースとなる文献 [2] の方法について説明する．

n 個の区分的線形抵抗を含む直流回路は，一般に次のような形の PLI 方程式で記述することができる．

$$f(x) \triangleq Pg(x) + Qx - r = 0 \qquad (1)$$

ただし，$x = (x_1, x_2, \cdots, x_n)^T \in \mathbb{R}^n$ は n 次元変数ベクトル，$g(x) = [g_1(x_1), g_2(x_2), \cdots, g_n(x_n)]^T$ は PLI 関数，P, Q は $n \times n$ 定数行列，r は n 次元定数ベクトルとする．

PLI 関数は，図 1(a) に示すような「2 つの区分的線形関数に挟まれた非凸多角形」で表され，図 1(b) に示すように各区間において「縦軸が平行な台形」からなる集合値写像となる．また，PLI 方程式の解は，一般に非連結な n 次元多面体の集合となる．

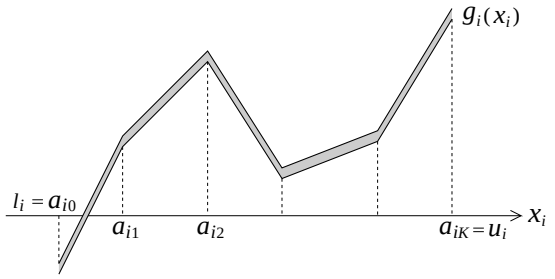

図 2　PLI 関数

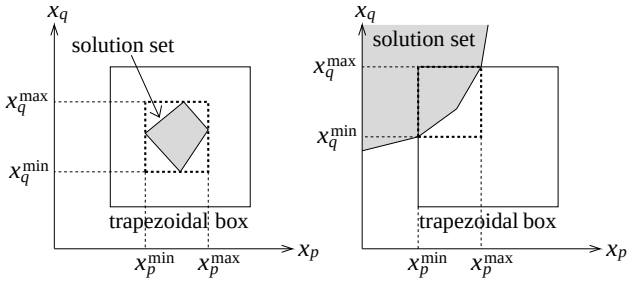

図 3　解集合の求め方

本章では，n 次元空間における直方体領域 $D = ([l_1, u_1], \cdots, [l_n.u_n])^T \in \mathbb{R}^n$ に存在する式 (1) の全ての解集合を求める問題を考える．簡単のため，PLI 関数 $g_i(x_i)$ は全て図 2 のような K 個の台形からなるものとし，f が台形となるような領域を台形領域と呼ぶ．

いま，変動領域を描画したい 2 次平面を $(x_p,\ x_q)$ 平面とする．ここで次のような混合整数計画問題を考える．

$$
\begin{aligned}
&\max/\min\ x_p \quad \text{and} \quad \max/\min\ x_q \\
&\text{subject to} \\
&\quad Py + Qx - r = 0 \\
&\quad x_i = a_{i0} + \sum_{j=1}^{K} \delta_{ij} \\
&\quad y_i \geq b_{i0}^L + \sum_{j=1}^{K} \frac{b_{ij}^L - b_{ij-1}^L}{a_{ij}^L - a_{ij-1}^L} \delta_{ij} \\
&\quad y_i \leq b_{i0}^U + \sum_{j=1}^{K} \frac{b_{ij}^U - b_{ij-1}^U}{a_{ij}^U - a_{ij-1}^U} \delta_{ij} \\
&\quad \Delta_{i1}\mu_{i1} \leq \delta_{i1} \leq \Delta_{i1} \\
&\qquad \vdots \\
&\quad \Delta_{ij-1}\mu_{ij-1} \leq \delta_{ij-1} \leq \Delta_{ij-1}\mu_{ij-2} \\
&\quad \Delta_{ij}\mu_{ij} \leq \delta_{ij} \leq \Delta_{ij}\mu_{ij-1} \\
&\quad \Delta_{ij+1}\mu_{ij+1} \leq \delta_{ij+1} \leq \Delta_{ij+1}\mu_{ij} \\
&\qquad \vdots \\
&\quad 0 \leq \delta_{iK} \leq \Delta_{iK}\mu_{iK-1}, \qquad i = 1, 2, \cdots, n
\end{aligned}
\tag{2}
$$

式 (2) では x_p, x_q を目的関数として最大化・最小化でそれぞれ 2 回ずつ解く．ここで，解プール機能を用いた CPLEX を適用する．これにより，解集合を含む全ての台形領域で式 (2) が解かれる．従って x_p, x_q を目的関数として最大化・最小化を行うと，解集合を含む台形領域で x_p, x_q がそれぞれ最大，最小となる点が得られる．この 4 つの解から，図 3 の点線で示すように，解集合を含む最小の n 次元直方体を求めることができる．

3.　提案手法

文献 [3] では，多次元空間上の複数の平面で解集合を描画したい場合は，式 (2) の目的関数を

$$\max/\min\ x_1,\ \max/\min\ x_2,\ \cdots,\ \max/\min\ x_n$$

に変更する．すなわち，式 (2) を $2n$ 回解くことで任意の (x_i, x_j) 平面の解集合を描画できると記されている．

しかし，混合整数計画問題を $2n$ 回も解くのは非効率的であり，加えて従来法には 2 つの不必要な計算が含まれている．一つは，二分木探索における最適化．台形領域ではないノードに対して x_i を最大化・最小化する必要ない．もう一つは，各台形領域上で $2n$ 回も行われる単体法のフェーズ I．台形領域上では制約条件は同じであるため，フェーズ I は 1 回行えば十分である．

そこで本論文では，次のような方法を提案する．提案手法では，まず次のような混合整数計画問題を考える．

$$
\begin{aligned}
&\max\ (\text{arbitrary constant}) \\
&\text{subject to} \\
&\quad Py + Qx - r = 0 \\
&\quad x_i = a_{i0} + \sum_{j=1}^{K} \delta_{ij} \\
&\quad y_i = b_{i0} + \sum_{j=1}^{K} \frac{b_{ij} - b_{ij-1}}{a_{ij} - a_{ij-1}} \delta_{ij} \\
&\quad \Delta_{i1}\mu_{i1} \leq \delta_{i1} \leq \Delta_{i1} \\
&\qquad \vdots \\
&\quad \Delta_{ij-1}\mu_{ij-1} \leq \delta_{ij-1} \leq \Delta_{ij-1}\mu_{ij-2} \\
&\quad \Delta_{ij}\mu_{ij} \leq \delta_{ij} \leq \Delta_{ij}\mu_{ij-1} \\
&\quad \Delta_{ij+1}\mu_{ij+1} \leq \delta_{ij+1} \leq \Delta_{ij+1}\mu_{ij} \\
&\qquad \vdots \\
&\quad 0 \leq \delta_{iK} \leq \Delta_{iK}\mu_{iK-1}, \qquad i = 1, 2, \cdots, n
\end{aligned}
\tag{3}
$$

Algorithm 1 Outline of one-tree algorithm: phase I

```
1: Preprocessing with only primal reductions
2: Set of open nodes : Nopen ← {rootnode}
3: Set of stored nodes : N_stored ← ∅
4: Objective value of the incumbent: z* ← +∞
5: while Nopen ≠ ∅ do
6:    Choose a node n from Nopen
7:    Solve LP at node n. Solution is x(n) with objective z(n).
8:    if z(n) ≥ z* then
9:       Fathom the node and keep it for phase II:
         Nopen ← Nopen \ {n}; N_stored ← N_stored ∪ {n}
10:   else
11:      if x(n) is integer-valued then
12:         x(n) becomes new incumbent:
            x* ← x(n); z* ← z(n)
13:         Fathom the node and keep it for phase II:
            Nopen ← Nopen \ {n}; N_stored ← N_stored ∪ {n}
14:      else
15:         Choose branching variable i such that x_i(n) is fractional
16:         Build children nodes n_1 = n ∩ {x_i ≤ ⌊x_i(n)⌋} and
            n_2 = n ∩ {x_i ≥ ⌊x_i(n)⌋ + 1}
17:         Nopen ← Nopen ∪ {n_1, n_2} \ {n}
18:      end if
19:   end if
20: end while
```

図 4　解プール機能のアルゴリズム（フェーズ I）

式 (3) を解プール機能を用いた CPLEX で 1 回だけ解く．これにより，解集合を含む全ての台形領域が μ_{ij} の組み合わせによって得られる．この場合，目的関数が任意の定数であるため，二分木探索の過程における最大化・最小化は行われず，各ノードで子問題の実行可能性だけが判定される．そのようにしてたどり着いた台形領域において，式 (3) は線形計画問題となる．従って，それらの台形領域上で線形計画問題を $2n$ 回解くことにより，解集合を包み込む最小の n 次元直方体を求めることができる．この場合，単体法のフェーズ I は 1 回行うだけでよく，その後双対単体法でフェーズ II のみを行う．これによりピボット演算回数は激減する．

式 (3) を 1 回解くだけで解集合を含む全ての台形領域が求まることを示す．CPLEX の解プール機能は文献 [4] のアルゴリズムを使用している [5, p.283]．このアルゴリズムの概略を図 4,5 に示す．

CPLEX の解プール機能は図 4,5 に示すように，2 つのフェーズからなるアルゴリズムである．図 4 では標準的な分枝限定法と同様に最適化を行うが，図 4 の 8,9 行目より，暫定解よりも悪い解となるノードも，図 5 のフェーズ II で準最適解を得るために保持される．

図 5 のフェーズ II では q% 以内の準最適解を得る．q はユーザが指定可能なパラメータで，本手法ではデフォルト値（1.0e+75）でよい．図 5 の 6,7 行目より，目的関数が任意の定数であるため条件式は常に偽となる．これにより，実行可能ノードは除去されずに図 5 の 12,13 行目で分枝される．従って，CPLEX の解プール機能を用いることにより実行可能ノードは除去されず解集合を含む全ての台形領域にたどり着くことが証明できる．

Algorithm 2 Outline of one-tree algorithm: phase II

```
1: Reuse tree from phase I: Nopen ← N_stored
2: Reuse incumbent from phase I: Set of solutions: S ← {x*}
3: while Nopen ≠ ∅ do
4:    Choose a node n from Nopen
5:    Solve LP at node n. Solution is x(n) with objective z(n).
6:    if z(n) > z* + q|z*|/100 then
7:       Fathom the node: Nopen ← Nopen \ {n}
8:    else
9:       if x(n) is integer-valued then
10:         x(n) is added to the pool of solutions if it is not a
            duplicate: if x(n) ∉ S, then S ← S ∪ {x(n)}
11:      end if
12:      Choose branching variable i such that it is not fixed by
         the local bounds of node n: lb_i(n) < ub_i(n)
13:      Build children nodes n_1 = n ∩ {x_i ≤ ⌊x_i(n)⌋} and
         n_2 = n ∩ {x_i ≥ ⌊x_i(n)⌋ + 1}
14:      Nopen ← Nopen ∪ {n_1, n_2} \ {n}
15:   end if
16: end while
```

図 5　解プール機能のアルゴリズム（フェーズ II）

4. 数値例

本章では数値実験結果をいくつか示し，提案したアルゴリズムの有効性を検証する．使用計算機は Hewlett-Packard Z440 (CPU: Intel Xeon Processor E5-1630 v3 3.70GHz)，整数計画ソルバーとしては CPLEX v12.6.3 を使用した．プログラミング言語は C++ である．

例 1: 文献 [6] の Example 5 で扱われている PLI 方程式を考える．ただし，$n = 500$, $K = 10$, 変動幅 $w = 0.2$ とする．この問題に対し，従来法である文献 [6] の方法と提案手法をそれぞれ適用した．

従来法の計算時間，即ち混合整数計画問題を 1000 回解くのに要した時間は 609,491 秒（約 7 日間）であった．

一方，提案手法の計算時間において混合整数計画問題を 1 回解くのに要した時間は 240 秒，線形計画問題を解くのに要した時間は 364 秒であった．従って，提案手法の計算時間は 604 秒であった．これは，提案手法が従来法よりも約 1000 倍高速であることを意味する．

例 2: 例 1 と同様の PLI 方程式を考える．ただし，

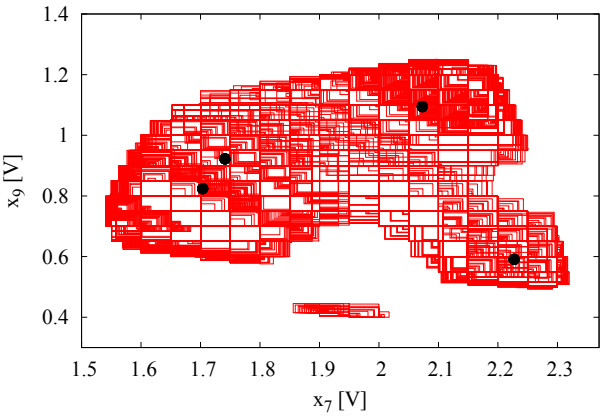

図 6　複雑な形状の解集合例（例 3）

$n = 100$, $K = 50$, $w = 0.2$ とし，初期領域を $D = ([-2,5],\cdots,[-2,5])^T$ とする．このとき，提案手法において混合整数計画問題を 1 回解くのに要した時間は 218 秒であった．また，得られた解集合を含む台形領域の数は 3,706 個であった．次に，線形計画問題を $2 \times 100 \times 3,706 = 741,200$ 回解く．計算時間は 455 秒であった．従って，提案手法の計算時間の合計は 673 秒であった．

例 3: 例 1,2 と同様の PLI 方程式を再度考える．ただし，$n = 10$, $K = 100, = w = 0.4$ とし，提案手法を適用した．このとき得られた (x_7, x_9) 平面上の解集合の一部を図 4 に示す．黒い点は $w = 0$ としたときの解を表している．図 4 のように変動を考慮したときの複雑な解集合を得られることが分かる．このとき，解集合の数は 16,272 個であり，計算時間は 62 秒であった．

例 4: 最後に，回路以外の例についても考える．文献 [6] で扱われている問題を考える．ただし，$n = 100$, $K = 20$, $w = 0.004$ とし，従来法と提案手法をそれぞれ適用した．このとき，従来法の計算時間は 15,440 秒であった．提案手法の計算時間は 79 秒であった．

全ての例題において，提案手法は従来法に比べて約 $2n$ 倍高速になっていた．これは，混合整数計画問題を解く回数が従来法に比べて $1/2n$ 倍になっており，また 3 章で示した不必要な計算が提案手法で改善されたためであると考えられる．

5. むすび

本論文では，整数計画法を用いた区分的線形区間方程式の全ての解集合を求める方法を提案した．本手法は混合整数計画問題を 1 回解き，次に線形計画問題を $2nB$ 回（B は混合整数計画問題を解いたときに得られる台形領域の数）解くことで PLI 方程式の全ての解集合を求めることができる．本手法もインプリメンテーションが容易で，従来法と比べて約 $2n$ 倍高速である．

文　献

[1] K. Yamamura, “Finding all solution sets of piecewise-trapezoidal equations described by set-valued functions,” Reliable Computing, vol. 9, no. 3, pp. 241–250, May 2003.

[2] K. Yamamura and N. Tamura, “Finding all solutions of separable systems of piecewise-linear equations using integer programming,” J. Computational and Applied Mathematics, vol.236, issue 11, pp.2844–2852, May 2012.

[3] K. Yamamura and S. Ishiguro, “Finding all solution sets of piecewise-linear interval equations using integer programming,” Reliable Computing, vol.23, pp.73–96, July 2016.

[4] E. Danna, M. Fenelon, Z. Gu, and R. Wunderling, “Generating multiple solutions for mixed integer programming problems,” in Integer Programming and Combinatorial Optimization, Proc. of 12th International IPCO Conference, Ithaca, NY, USA, June 25–27, 2007, pp.280–294, Springer-Verlag, Berlin, Heidelberg, 2007.

[5] IBM, IBM ILOG CPLEX Optimization Studio, CPLEX User’s Manual, Version 12, Release 6, http://www.ibm.com/support/knowledgecenter/SSSA5P 12.6.3/ilog.odms.studio.help/pdf/usrcplex.pdf.

[6] K. Yamamura, K. Suda, and N. Tamura, “LP narrowing: A new strategy for finding all solutions of nonlinear equations,” Applied Mathematics and Computation, vol.215, no.1, pp.405–413, Sep. 2009.

研 究 業 績

[1] K. Yamamura, H. Takahara, and Y. Takane, “Finding all solution sets of piecewise-linear interval equations using integer programming,” Proc. 2017 IEEE Workshop on Nonlinear Circuit Networks, pp.78–81, Dec. 2017.

[2] K. Yamamura and Y. Takane, “An efficient method for finding all characteristic curves of piecewise-linear resistive circuits using integer programming,” Proc. 2018 IEEE Workshop on Nonlinear Circuit Networks, pp.66–69, Dec. 2018.