# 二次元多重連結領域上の構造安定な非圧縮流の木表現の可視化手法

2016SE024 亀谷拓磨　2016SE076 田島嘉人　2016SE090 渡辺康平
指導教員：横山哲郎

## 1　はじめに

流れをそのトポロジーに注目して解析する手法が知られている．トポロジカルには，連続的に変形できる図形は同じ形とみなされる．例えば，四角形と三角形は同じ形である．トポロジーに着目することは，詳細な情報は無視することになるが，データの本質的な特徴を見つけやすくなるなどのメリットがある．近年では，ビッグデータの解析などをトポロジーに着目して行うトポロジカルデータアナリシス（TDA）と呼ばれる解析手法が注目されている [1].

TDA は流体力学の分野においても有効な解析であることが知られている．トポロジカルフローデータアナリシスは流れをトポロジカルな観点で解析する．トポロジカルな観点に着目することは大域的な構造に着目するということを意味し，これにより流れの本質的な構造を抜き出すことが可能となる．この解析法の代表的なアプローチとして，流体をその構造安定性に着目して解析する方法がある．構造安定性は，力学系に小さな乱れが加わっても流れのトポロジーが変化しない性質のことである．構造安定性は一般の流体においては成立しないが，文献 [2] によれば，有限の精度や有限の時間における挙動に着目したり，本質的な構造に着目するなどの現実的な目的の範囲内であれば，現実の流体にもこの考え方が適用できる．構造安定性に着目すれば，構造安定性から流れの変化を考察することができるようになる．例えば，語表現の研究では流れのトポロジーに対応した文字列を定義することで，流体構造変化を文字列の変化として考察することが可能となった [3].

本研究では，流れのトポロジーの表現の一つである木表現で表された流線を 2 次元上の表現に変換する手法を考案する．木表現は流線構造を代数的に扱い流れの解析を行うことができるが，その表現から直観的に 2 次元上の流れの形状を把握することは困難である．そのため，2 次元の形状を得たい場合は解析者がその都度木表現を組み合わせて図示化することになるが，木表現が複雑になればなるほどその変換も煩雑になり，ともすれば途中で間違った変換を行ってしまうこともあり得る．また，著作物などに木表現の図を挿入しようとする場合，描画ソフトを用いてそれを手作業で描く必要がある．本研究は木表現を図に自動的に変換する方法を与えることで先述したような手間を省き，効率的で正確な木文法による流れの解析を実現するものである．

## 2　関連研究

全ての 2 次元多重連結領域上の構造安定な非圧縮流は，定められた初期構造に対して軌道と特異点をある規則に従い帰納的に組み合わせることで表現できる．流れの木表現は，この規則を木文法で表した流れの表現方法である．

本研究において木表現は [4] により与えられた木文法を用いる．この木文法は，$G = (S, N, F, R)$ によって定められる．$S$ は開始記号，$N$ は非終端記号の集合，$F$ は終端記号の集合，$R$ は生成規則である．ここで, $N = \{S, A_\emptyset, B_+, B_-, C_+, C_-, C_+^*, C_-^*\}$, $F = \{a_\emptyset, b_{\emptyset+}(,\{\}), b_{\emptyset-}(,\{\}), a_+(), a_-(), a_2(,), b_{++}\{,\}, b_{+-}\{,\}, b_{--}\{,\}, b_{-+}\{,\}, \beta_+\{\}, \beta_-\{\}, c_+(,), c_-(,), l, n, cons(,)\}$ とする．生成規則 $R$ は以下のように表される．

$$
\begin{array}{lcl}
S & \rightarrow & a_\emptyset(A^*) \mid b_{\emptyset+}(B_+,\{C_-^*\}) \mid b_{\emptyset-}(B_-,\{C_+^*\}) \\
A & \rightarrow & a_+(B_+) \mid a_-(B_-) \mid a_2(C_+^*,C_-^*) \\
A^* & \rightarrow & n \mid cons(A,A^*) \\
B_+ & \rightarrow & l \mid b_{++}\{B_+,B_+\} \mid b_{+-}(B_+,B_-) \mid \beta_+\{C_+^*\} \\
B_- & \rightarrow & l \mid b_{--}\{B_-,B_-\} \mid b_{-+}(B_-,B_+) \mid \beta_-\{C_-^*\} \\
C_+ & \rightarrow & c_+(B_+,C_-^*) \\
C_- & \rightarrow & c_-(B_-,C_+^*) \\
C_+^* & \rightarrow & n \mid cons(C_+,C_+^*) \\
C_-^* & \rightarrow & n \mid cons(C_-,C_-^*)
\end{array}
$$

以上の木文法によって生成された木は，流線の形状を表す．開始記号 $S$ は 3 つの基本パターン $a_\emptyset$，$b_{\emptyset+}$，$b_{\emptyset-}$ からなり，これは流れの初期構造に対応する．この基本パターンを図 1 に示す．$a_\emptyset$ は一様流を表し，$b_{\emptyset+}$，$b_{\emptyset-}$ はともに円盤状の流れで最外境界部をもつ流線を表す．また，木文法では + が反時計回り，− が時計回りの流れを表す．これらの初期構造に A 系，B+ 系，B − 系，C+ 系，C − 系の流れを生成規則に従い組み合わせる．$A$ 系，$B_+$ 系，$B_-$ 系，$C_+$ 系，$C_-$ 系の流れをそれぞれ図 2–6 に示す．

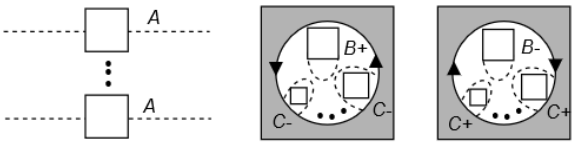


図 1　3 つの基本パターン　左から $a_\emptyset$，$b_{\emptyset+}$，$b_{\emptyset-}$

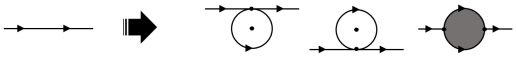

図 2　$A$ 系の流れ

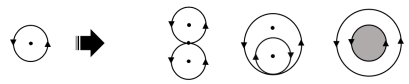

図 3　$B_+$ 系の流れ

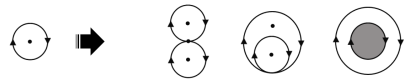

図 4　$B_-$ 系の流れ

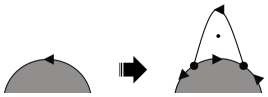

図 5　$C_+$ 系の流れ

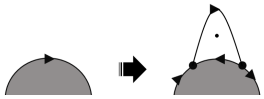

図 6　$C_-$ 系の流れ

## 3　自動可視化手法の仕様

### 3.1　図が満たすべき条件

本研究では，木表現から 2 次元上のトポロジーへの自動変換手法を考案し，実装する．この節では，実装したプログラムがどのような図を作図すれば本研究の目的を達成できたと言えるか定める．本研究では以下の 3 つの指標を満たすことができれば，プログラムの利用者によって見やすくトポロジーの検証に便利であり，本研究の目的を満たすことができたプログラムであるとする．

- 線が重ならず交差しない
- 線の間が適切な距離を保つ
- 線が滑らかである

実装したプログラムは少なくとも木表現を 2 次元上のトポロジーに変換した際に，その両方が確実に同一のものを再現している必要がある．例えば，木表現が $a_\emptyset(cons(a_+(b_{++}\{l,l\},l),n))$ であれば，2 次元上のトポロジーも $a_\emptyset(cons(a_+(b_{++}\{l,l\},l),n))$ を表していなければならず，間違っても $a_\emptyset(cons(a_-(b_{--}\{l,l\},l),n))$ ではならない．線が重なったり交差してしまうと，それは期待するトポロジーとは異なるものを示す．

トポロジーを表す線同士の適切な距離は，主観的であるが，あまりに離れているように感じられると図の見栄えが悪くなる．逆に，近づきすぎていても同様である．

線の滑らかさも線同士の距離に同じく，角ばっていたりなどするとトポロジーの判定には影響が出ないが，見栄えが悪くなる．

### 3.2　流線の表現方法

各トポロジーを 2 次元上の図として再現するためには，各トポロジーを 2 次元上に表現可能な図形でモデル化する必要がある．$A$ 系や $B_+$ 系，$B_-$ 系を構成する流線と物理境界は円形であるため，円でモデル化できる．したがって，これらで構成されているすべての $A$ 系と $B_+$ 系，$B_-$ 系のトポロジーは円を組み合わせることで表現できる．しかし，$C_+$ 系，$C_-$ 系のトポロジーは基本的な図形の組み合わせで表現することは難しい．そこで，$C_+$ 系，$C_-$ 系はそのトポロジーを図 7 のように点でモデル化し，この点を結ぶような滑らかな線を描画することで表現する．

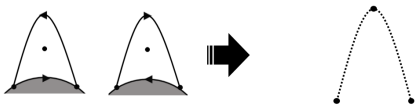

図 7　$C$ 系のモデル化

### 3.3　整合性のとれた図を作図するアルゴリズム

自動描画で木表現のトポロジーを 2 次元上に再現するためには，それぞれのトポロジーの形を正確に描画することも求められるが，さらにすべてのトポロジーの位置関係の整合性を保つ必要がある．そこで，我々は葉から木表現をたどり大きさを決定し，描画は根から行う方法を考えた．この方法では，描画の位置を決めずにすべての部分木に対応する流線図の大きさのみを決定し，その大きさをもとに根から位置を決定し描画していくことで位置と大きさの整合性をとれた図を作図できる．ここで，このアルゴリズムを実際にプログラムにするにあたって子ノードから親ノードに正確に大きさを伝達する方法を考える必要がある．そこで，これを伝達する方法としてすべてのトポロジーに画一的な指標を定義する．これを占有領域と呼ぶ．この占有領域を子ノードから親ノードに伝達し親ノードの占有領域を決定する．占有領域は円で表す．これは，ほとんどのトポロジーが図 8 のように円状で表すことが容易だからである．ただし，例外として $C$ 系のみは円状で表すことが難しい．そこで $C$ 系の占有領域は図 9 のような長方形とする．

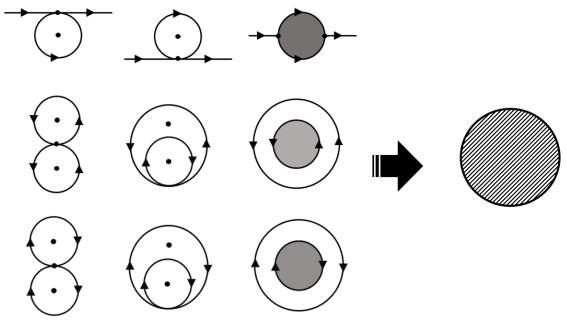

図 8　円で表す占有領域

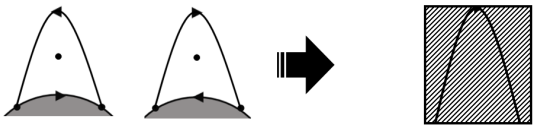

図 9　$C$ 系を表す占有領域

占有領域を用いた大きさ決定の例として，図 10–12 にそれぞれ $b_{++}$，$c_{+}$，$\beta_{+}$ の大きさ決定方法を示す．それぞれの図中のグレーの部分は，子ノードの占有領域を表している．また，図 10，図 12 中の $r$ はそれ自身の占有領域を表す円の半径，図 11 中の $w$，$h$ はそれぞれ占有領域の底辺長と高さであり，これらが自身の占有領域の大きさを表す情報となる．図 12 中の $R$ は物理境界を表す円の半径である．この値は物理境界に接する $C$ 系の占有領域の底辺長の合計で求められた円周から決まる．これを表した図が図 13 である．

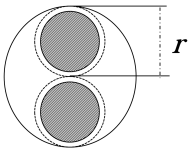


図 10 $b_{++}$ の占有領域

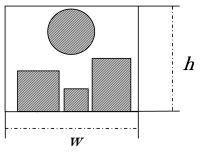


図 11 $c_{+}$ の占有領域

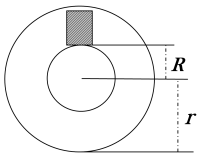


図 12 $\beta_{+}$ の占有領域

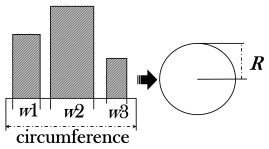


図 13 物理境界

描画時には占有領域を用いて決定された各流線図の大きさをもとに根から描画を行う．描画の流れを表した手順が図 14 である．占有領域を円で表す流線は，親ノードから占有領域の中心の座標を得ることで，その座標をもとに描画できる．占有領域が長方形である $C$ 系の流線は親の物理境界に接するような描画角度を考慮する必要があるため占有領域の中心の座標では描画できない．そのため，$C$ 系の流線はそれが接する物理境界の情報を利用し，三角関数を用いて流線を構成する点の座標を決定し描画する．例として，$b_{++}$ と $C$ 系を描画する処理をそれぞれ図 15，図 16 に示す．図 15 中の $R1$，$R2$ はそれぞれ対応する環状の流線を表す円の半径であり，$center$ は親ノードのインスタンスで決められたこの $b_{++}$ の占有領域の中心の座標である．$head\_center$，$tail\_center$ はそれぞれ対応する子ノードの占有領域の中心の座標を表し，これが子ノードの描画時に用いられる．図 16 中の $point1$ は同じ物理境界に接する他の $C$ 系と描画時に重なったりしないように親ノードで決定される．この $point1$ から円周上で自身の占有領域の底辺長分ずらした点が $point2$ である．$point3$ は $\theta_2 - \theta_1$ の角度で，かつ，円周上から自身の占有領域の高さ分離れた座標である．

### 3.4 描画アルゴリズム実装に適したクラス図

作成するプログラムが木表現の入力から 2 次元上のトポロジーを描画する処理上，このプログラムでは木表現

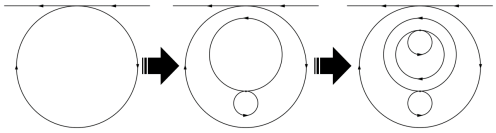

図 14 描画の流れ

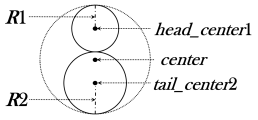


図 15 $b_{++}$ 描画のイメージ

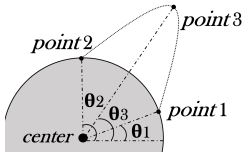


図 16 $C$ 系描画のイメージ

の入力に対して字句解析と構文解析を行い，構文木を作成する必要がある．また，作成した構文木に則った処理を行う必要がある．このような処理を行うのに適したデザインパターンにインタプリタパターンがある．インタプリタパターンは，形式的に記述された記号列を，解析した結果に則って処理したい場合に利用されるデザインパターンである [5]．インタプリタパターンに基づいて作成したクラス図の一部が図 17 である．

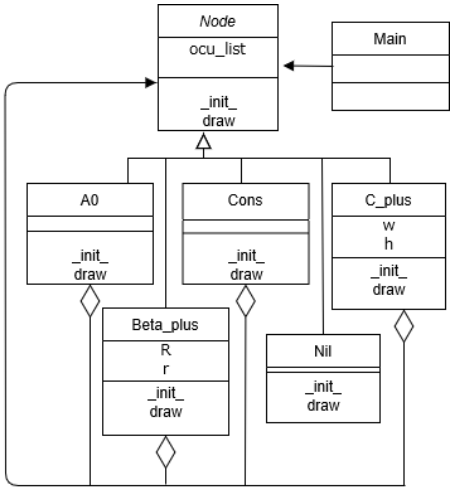


図 17 クラス図 (一部抜粋)

実際のクラス図は，木文法の非終端記号と終端記号に対応しそれぞれの流線を描画する処理を行う 17 のクラスと，これら 17 のクラスに共通のインタフェースを定義する Node クラス，入力された木表現を解析し構文木を生成する Main クラスの 19 のクラスで構成される．

抽象クラスである Node クラスには，コンストラクタメソッドである init メソッドと draw メソッドを定義する．また，各非終端記号と終端記号に対応するクラスではこれらのメソッドをオーバライドし，init メソッドでは抽象構文木の生成と同時に占有領域の大きさの決定を，draw メソッドでは描画処理を行う．また，構文木を作成する都合上，処理結果と処理対象を同一視する必要があるため，

Node クラスと非終端記号に対応するクラスには集約関係がある．

例として $\beta_+$, $C_+$ に対応する Beta_plus, C_plus クラスを挙げる．Beta_plus クラスはインスタンス変数に $R$ と $r$ をもつ．この変数はそれぞれ図 12 中の $R$, $r$ に対応する．init メソッドでは，子ノードの占有領域の大きさを表すインスタンス変数の値を取得し，それをもとに自身の占有領域の大きさを表す $R$ と r を決定する．draw メソッドは，自身を描画すべき座標を引数にとり，これを描画する．また，非終端記号の draw メソッド内では子ノードの draw メソッドを呼び出し，子ノードが描かれるべき座標の情報を引数として与える．このように，draw メソッドを連鎖的に呼び出すことで，再帰的に流線の描画を行う．C_plus クラスはインスタンス変数に $w$ と $h$ を持つ．この変数は Beta_plus と同様に，図 11 中の $w$, $h$ に対応し，init メソッドで決定される自身の占有領域の大きさを保持するためのインスタンス変数である．また，draw メソッドでは前述した図 16 の処理を行い，子ノードの draw メソッドを呼ぶ．

流線を描画しない終端記号に，木を連結する役割をもつ *cons* がある．これに対応する Cons クラスは init メソッドでは親ノードの占有領域の大きさを決定するために Cons クラス自身の子ノードの占有領域の情報を親ノードに，draw メソッドでは親ノードで決められた子孫ノードの描画位置の情報をその子ノードに伝達する役割をもつ．具体的には，init メソッドでは子ノードの占有領域の大きさをリストとしてまとめ *ocu_list* に格納し，draw メソッドでは親ノードで決められた子孫ノードを描画する位置をまとめたリストを子ノードに分配する．

## 4 実装

3.3 節のアルゴリズムと 3.4 節のクラス図をプログラミング言語 Python とグラフ描画ライブラリ Matplotlib，構文解析器 PLY を用い実装した．また，$C_+$ 系，$C_-$ 系の流線の補間には，一般的な補間方法であるスプライン補間を用いた．このプログラムで $a_\emptyset(cons(a_-(b_{--}(b_{-+}(l,\ l),\ l)),\ n))$ と $b_{\emptyset+}(b_{+-}(b_{++}(l,\ l),\ l),\ (cons(c_-(l,\ n),\ cons(c_-(l,\ n),\ n))))$ を変換した結果がそれぞれ図 18 と図 19 である．

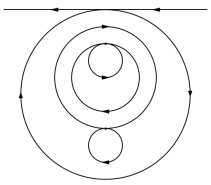

図 18 実行例 1

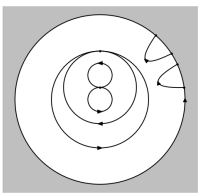

図 19 実行例 2

## 5 おわりに

本研究では，木表現で表された構造安定な非圧縮流を 2 次元上のトポロジー表現へ変換するアルゴリズムを設計した．また，そのアルゴリズムに適したクラス図を設計し，これを Python を用いて実装した．設計した描画アルゴリズムは，占有領域という概念を用いて流線図の大きさを葉から順に大きくすることによって流線間の距離を一定に保つことができ，線を必ず重ならせない．また，それぞれの流線を円か点でモデル化することによって滑らかな線を表現した．以上より，考案したアルゴリズムは 3.1 節で定めた条件を満たした図を，下で述べる欠点以外では作成できるものであると言える．

実装したプログラムにより，木表現で表されたトポロジーを速やかに 2 次元上のトポロジーへ間違いなく正確に変換することが可能となった．実際，約 4000 の流れの木表現を本プログラムで変換するテストを行ったところ，全て正確に 2 次元の図が表示された．したがって，本アルゴリズムとプログラムを用いることで，流体解析者が木表現で表現されたトポロジーを，少なくともテストケースの範囲内では間違いなく速やかに直観的に理解できるようになったと言える．

しかし，3.1 節で定義した図の条件を完全に満たすことができたかどうかについては検討の余地があると考える．本研究で製作したアルゴリズムの占有領域は設計を単純化するため，各流線を覆う円で表されている．これは，$b_{++}$ なども 1 つの円として表しているため，実際の流線よりも占有領域の面積が大きい．このため，$b_{++}$ など実際の流線よりもその占有領域の面積が大きい流線が増えれば増えるほど空白が多くなり，一見して無駄な部分が多く見づらい．したがって，適切な距離を完全に保てているとは言い難い．解決には，占有領域をより柔軟に定義するアルゴリズムの設計が求められる．

## 参考文献


[1] 梅田裕平： データの形が教えてくれること-トポロジカル・データ・アナリシスとその応用-， 情報処理，Vol.57，No.11，pp.1122–1127 (2016).

[2] 荒井迅：トポロジカルな流れ構造の理解へ向けて，ながれ，Vol.33，No.1，pp.23–28 (2014).

[3] Yokoyama, T. and Sakajo, T.: Word representation of streamline topologies for structurally stable vortex flows in multiply connected domains. *Proc. Roy. Soc. A*, Vol.469, No.2150, pp.1–18 (2013).

[4] 加藤舞，内藤綾香，横山哲郎，横山知郎：円盤上の非圧縮流の反転の解析，情報処理学会　第 81 回全国大会講演論文集，Vol.1900，pp.319–120 (2019).

[5] 結城浩：Java 言語で学ぶデザインパターン入門，SB クリエイティブ (2001).