

二次元多重連結領域内における構造安定な非圧縮流れ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

流体力学は、気体や液体の運動について取り扱う力学の主要な研究分野である。流体力学の歴史は古く、この学問が確立される以前から人々の生活に即したものとして発展してきた。流体解析の手法の1つに離散解析がある。離散解析は流れをトポロジカルな観点から解析する。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。つまり流れの大枠に着目した解析を行いたい場合、離散解析を用いることで効率的に流れの解析を進められるということである。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。トポロジーは位相幾何学の用語であり、この立場に立てば、連続的に変形できる図形は同じ形である。例えば、四角形と三角形は同じ形である。構造安定性は一般の流体にはあり得ないが、文献[1]によれば、有限の制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応できる。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して対応した文字列を定義することで、2つの流体構造の中間構造を文字列の変化として考察することが可能となった[2]。

本研究は、離散解析手法の表現の1つである木表現に対して、図上への可視化手法を与える。ここで、本研究で扱う木表現は[3]によって与えられた表現方法である。また、以下で木表現とはこの表現方法を指す。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えた変換を行ってしまうこともあり得る。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、解析者の効率的で確実な木文法による流れの解析に寄与するものである。

2 関連研究

本研究で扱う木表現は、語表現の研究を発展させたものである。語表現の研究では、流線のトポロジカルな構造を分類し、それぞれの流線構造を系統的に文字列で表現する

アルゴリズムを与えた[2]。語表現を用いた例として、翼の揚抗比の時間変化を語表現によって表した研究がある[4]。語表現は、有界な多重連結領域上で非圧縮かつ非粘性で構造安定な流れを代数的に表すことができる。多重連結領域とは複数の障害物が含まれている領域のことである。ここでは非圧縮・非粘性という理想流体を仮定しているが、この仮定は現実の流体に対しての直接の適用に制限を与える。しかし、現実への応用に関しては文献[4]でその適用方法が考えられている。語表現の利点として、流線構造を数学的に厳密に分類できるようになるため、流れの変動を特徴付けて捉えることができるようになる点が挙げられる。また、流れを代数的に扱うことができるため、流線の構造の特徴を説明するための共通言語として用いることができる。文献[2]によって与えられた語表現には同じ流線に複数の語表現を与えることができるという問題があった。そのため、文献[5]によって自然な語表現を与えるアルゴリズムが与えられた。また、文献[6]によって流れの向きを考慮した場合の語表現を与えるアルゴリズムが与えられた。

木表現もその前提条件は語表現と共通であり、有界な多重連結領域上で非圧縮性かつ非粘性で構造安定である。木表現はその特性から、語表現より細かい流れの分類を可能とする。そのため流れの特徴を語表現より、多く捉えることを期待されている。実際、円盤状の非圧縮流の反転の解析を木表現によって行った研究では、語表現と比べた木表現の表現力の高さが確かめられた[7]。

語表現および木表現と関連のある離散解析手法として、コンレイ・モース分解、グラフクラスタリングがある[1]。これらの手法は、流体に対してそれぞれ異なるアプローチを持つが、流体をその構造安定性に着目して解析するという点では一致している。

3 自動可視化への準備

3.1 可読性の定義

作成するプログラムには、木表現で作成されたトポロジーを確実に図に再現することが求められるが、さらに、再現されたトポロジーは可読性が高いものである必要もある。本研究では、可読性の高さを定義する上でグラフ描画アルゴリズムにおける可読性の高さの基準[8]を参考にプログラムに求められる可読性を定義した。

- 線が重ならない
- 線の間が適切な距離を保つ
- 線が滑らかである

3.2 文法の定義

本文法 $G=(S, N, F, R)$ は以下のように定められる. S は開始記号, V は非終端記号の集合, F は終端記号の集合, R を生成規則とする. このとき, $N = \{ S, A, B_+, B_-, C_+, C_-, C_+^*, C_-^*, \}$, $F = \{ a_\phi(), b_\phi(\{\}), b_{\phi-}(\{\}), a_+(), a_-(), a_2(), b_{++}(\{\}), b_{+-}(\{\}), b_{--}(\{\}), b_{-+}(\{\}), \beta_+(\{\}), \beta_-(\{\}), c_+(\cdot), c_-(\cdot), l, \lambda, \text{cons}(\cdot, \cdot) \}$ とする.

流れの生成規則 R

$S \rightarrow a_\phi(A^*) \mid b_{\phi+}(B_+, \{C_-^*\}) \mid b_{\phi-}(B_-, \{C_+^*\})$

$A \rightarrow a_+(B_+) \mid a_-(B_-) \mid a_2(C_+^*, C_-^*)$

$A^* \rightarrow \lambda \mid \text{cons}(A, A^*)$

$B_+ \rightarrow l \mid b_{++}\{B_+, B_+\} \mid b_{+-}\{B_+, B_-\} \mid \beta_+\{C_+^*\}$

$B_- \rightarrow l \mid b_{--}\{B_-, B_-\} \mid b_{-+}\{B_-, B_+\} \mid \beta_-\{C_-^*\}$

$C_+ \rightarrow c_+(B_+, C_-^*)$

$C_- \rightarrow c_-(B_-, C_+^*)$

$C_+^* \rightarrow \lambda \mid \text{cons}(C_+, C_+^*)$

$C_-^* \rightarrow \lambda \mid \text{cons}(C_-, C_-^*)$

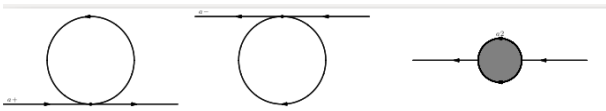


図1 a系の流れ 左から a_+ , a_- , a_2

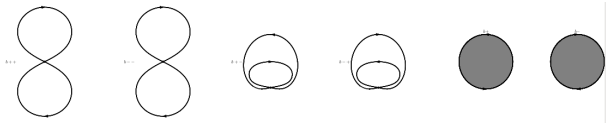


図2 b系の流れ 左から b_{++} , b_{--} , b_{+-} , b_{-+} , b_+ , b_-

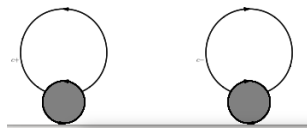


図3 c系の流れ 左から c_+ , c_-

3.3 アプローチ

本研究の目的の達成には, 3.1 の定義を満たすトポロジ的な流れの図を二次元上の画面に表示することを必須としている. アプローチとしては, コンピュータのディスプレイ上に作成した図を可視化し, 画像ファイルとして保存することを可能にするプログラミング言語を本研究で使った. 私達は様々な分野での実績があり多様な環境での信頼性が高い Python とベクタ形式での描画を支援する Asyptote の2つプログラミング言語をアプローチ方法とした.

3.4 構文解析

本研究では Python での実装を行うため, Python 専用の構文解析ライブラリ PLY を利用し構文解析を行った. PLY とは, Python Lex-Yacc の略であり, Python 専用の構文解析ライブラリである. lex.py と yacc.py からなる. lex.py では, 入力されたテキストを正規表現により定義されたデータ構造に分解し, 字句解析を行う. その後, yacc.py により, 自由文法によって定義された構文を評価して抽象構文木を作成し構文解析を行う.

4 実装

4.1 Python による実装

4.1.1 実装するクラス図

本研究では, デザインパターンの1つであるインタプリタパターンを利用してクラス図を作成した. インタプリタパターンについては, [9] を参考にした. インタプリタパターンは, 解析された結果に則り, 処理を実行したい場合に利用される. 本研究では, 与えられた文字列を PLY を利用し字句解析・構文解析を行う. そのため, 本文法を扱うのに適しているインタプリタパターンを利用し, クラス図を作成した. メリットとして, プログラムの修正を容易にし, 見た目をシンプルにすることができる. 今回作成したクラスが多くなったため, 例として一部分のみを図示する.

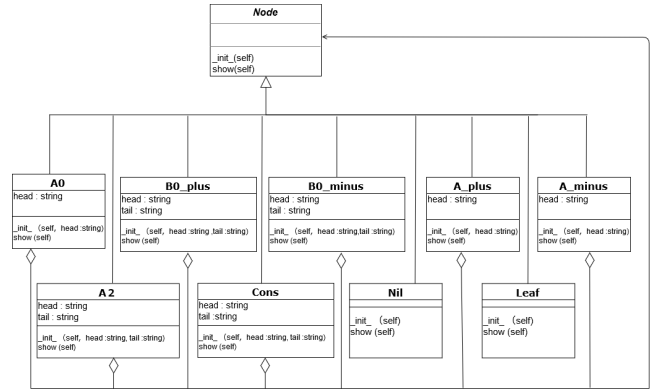


図4 作成したクラス図

4.1.2 描画機能の実装: 大きさの決定

Python で流線図を描画するにあたり, Python 用のライブラリである matplotlib を使用する. matplotlib はグラフ描画のためのライブラリであり, 棒グラフや折れ線グラフなどの一般的なグラフを作図することができる. また, それ以外にも平面上への関数・図形の描画や波のシミュレーションなど様々な描画にも利用できる.

自動描画を実現するプログラムは木表現を与えるのみで動作することが求められ, 私達が引数を与えて円を描くことはできない. また, 子を持つ親を描画する際, 子は親の

描く流線の内部に描画される必要があるため、必然的に親の大きさは子の大きさよりも大きくなる。そのため、プログラムには描画した際の大きさを柔軟に変更できる仕組みを組み込む必要がある。本研究では、この課題を解決するため木表現の葉から大きさを決定することにした。例として、 $a_+(b_{++}(b_{++}(l, l), l))$ という木表現を挙げる。この木表現では、 a_+ が b_{++} という子を持ち、その子である b_{++} は $(b_{++}(l, l), l)$ という 2 つの子を持っている親である。この場合、プログラムは $b_{++}(l, l)$ からその大きさを決める。続いて、 $b_{++}(l, l)$ の親である $b_{++}(b_{++}(l, l), l)$ の大きさを決定する。ここで、親の b_{++} の大きさは、子の $b_{++}(l, l)$ を踏まえた大きさとなる。そして最後に、その大きさをもとに、 a_+ の大きさが決定される。この流れを図 5 に示した。ここで、それぞれについて「大きさを決定する」としているのは、大きさを決定した時点では matplotlib のグラフ上に描画しないからである。これは、子から描画しようとしても、その時点ではグラフ中の位置を決定できないことによる。この例では、 $b_{++}(l, l)$ の位置を先に決定してしまうと、その親である b_{++} の位置もそれに依存して決まってしまう。しかし、もし親の b_{++} がもう一つの子を持っていた場合、葉から描画していく仕様上もう一つの子も既に決められた位置を持つことになる。ここで、子の位置を決定する時点で他の流線との位置関係を決定できる仕組みがあり、2 つの子の位置の整合性が取れているのであれば子から順に描画することもできるが、子の情報だけでは親との位置関係は分からないためこの仕組みは作成できない。そのため、実際にグラフ上に描画する前に根までのすべての大きさを決定する処理を行い、次に根の位置を決定し順に子を描画していく方法を取った。

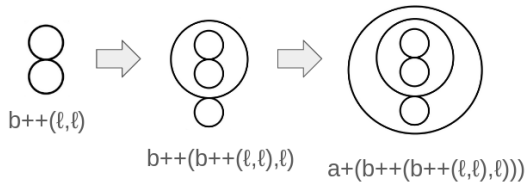


図 5 $a_+(b_{++}(b_{++}(l, l), l))$ 大きさ決定のイメージ

4.1.3 描画機能の実装：流線の描画

matplotlib では、円や長方形などの図形をもともと用意されている関数で呼び出したりできるが、これらは決められた形にしかならない。流線を描画するにあたり、一部のトポロジーは円などの固定された図形でも表現可能だが、 b 系を含む流線はそれらで表すことができず、時々に合わせて形を変えることが求められる。そこで、本研究ではこれらの b 系を含む図形を点でモデル化することにした。例として、 a_+ が子に b 系を持っていると想定した場合の流線を図 6 に示す。ただ、これは製作途中の仮のものであるため、その子の大きさの円を仮に与え描画している。この図形は、点 5 つから構成されている。子の専有領域の接合

部である 2 点と最右点・最左点・最下点の 3 点である。この 5 点を、スプライン補間で補間することにより流線を表現した。



図 6 b 系を持つ図形の作図

4.2 Asymptote による実装

Asymptote では、木表現に対しての 3.1 の定義を満たす図を作成するためのライブラリを提供し、実際に存在する流体をトポロジー的に再現することができれば本研究の目的を満たすことを示す。

4.2.1 提供するライブラリ内の関数

```
void ap(real d, pair s)
void am(real d, pair s)
void a2(real d, pair s)
void bpp(real d, pair s)
void bmm(real d, pair s)
void bp(real d, pair s)
void bm(real d, pair s)
void bpm(real d, pair s)
void bmp(real d, pair s)
void cp(real d, pair s)
void cm(real d, pair s)
```

以上が本文法の生成規則に準じた主な流れの関数である。だが、描画する際に描画対象の図によって流れの向きを調節する必要があるため、図が右向き (Right) か左向き (Left)、上向き (Up) か下向き (Down) の頭文字を関数の後ろにつける。例えば、右向きかつ上向きの a_+ の流れを描画するには `void apRU(real d, pair s)` となる。real 型 d は流線図の大きさを表し、pair 型 s は複素数 (x, y) であり流線図を場所を表す。これらの関数を組み合わせ、パラメータを調節することにより目的の流れを描画できる。

4.2.2 Asymptote による実行方法

Asymptote ではプログラムを実行する方法として、Ubuntu の端末や Windows のコマンドプロンプト上に直接コードを記述する interactive mode(対話モード) という方法が存在する。そこでライブラリを読み込み、画面上に表示させたい流線図に対応する関数を記述することで、画面上にトポロジー化された流線図を表示させる。

4.2.3 実行例

本研究では非圧縮流体を題材としている。しかし、トポロジー的な流れの図を描画できるかは、非圧縮流体でなくても実証は可能である。さらに、非圧縮流体以外の実際に存在する流れを例題に示すことで本研究の多様性が高まる

などが期待できる。本研究では、カルマン渦を実例として挙げた*1。



図7 カルマン渦

ある範囲の速度の流れの中に円筒状の障害物が存在すると、その後流側に規則正しく左右交互に並んだ2列の渦が発生する。この渦をカルマン渦と呼ぶ。本研究では、一部のカルマン渦をトポロジ的に表した図を Asymptote で再現する。トポロジ化してあるカルマン渦については、文献 [10] を参照した。



図8 トポロジー化したカルマン渦

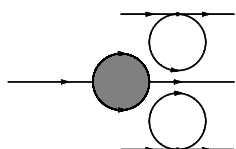


図9 作成したカルマン渦

図8は a 系と c 系で構成されてる。本研究の現状では c 系を組み合わせるの描画は実現できていないため、関数 amRU, amRD, a2R にパラメータを与えてカルマン渦を作図した。

5 おわりに

5.1 Python

作成中のプログラムは、現在再現できる a 系と b 系の組み合わせの範囲内では 3.1 により定められた条件を満たす図を製図できる。また、まだ再現できない系の流線に対しても葉から大きさを決定し根から描画する考え方は適用可能であると考え。しかし、一部の流れにはループが存在し、その部分の要素は無限に増やすことが可能である。そのため、要素数が無限に増えても柔軟に生成される図を変更できることが求められる。現在、これを解決するためグラフ描画アルゴリズムを利用を検討している。グラフ描画アルゴリズムを用いることで、文献 [8], [11] にあるように柔軟に図の変更を行うことができるのではないかと考えている。具体的には C 系の中で発生する流れ同士の距離感を一定に保つことなどを考えているが、まだ実装には至っていない。今後は、これらの描画手法を実装し、また、作成したプログラムの有用性についての検証を行ってきたい。

5.2 Asymptote

Asymptote で作成したプログラムも Python と同等な理由で、 c 系を組み合わせ可能な段階には至っていない。また、本研究の目的である入力を完全な本文法での描画は現状実装できなかった。今後は、本研究の目的達成のために改良を施していきたい。

参考文献

- [1] 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).
- [2] 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).
- [3] 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).
- [4] 横山知郎, 坂上貴之, 澤村陽一：二次元多重連結領域内における構造安定な非圧縮流れの文字列表現アルゴリズム, 数理解析研究所講究録, Vol.J101-D, pp.11–25(2014).
- [5] 横山哲郎, 横山知郎：ハミルトン曲面流に対応する語の列挙アルゴリズム, 電子情報通信学会論文誌 D, Vol.100, No.10, pp.892–894(2017).
- [6] 横山哲郎, 横山知郎：ハミルトン曲面流に対応する流れの向きを考慮した極大語の列挙アルゴリズム, 電子情報通信学会論文誌 D, Vol.101, No.8, pp.1220–1222(2018).
- [7] 加藤舞, 内藤綾香：多重連結領域上の安定非圧縮流の解析, 南山大学 2018 年度卒業論文 (2019).
- [8] 土井淳, 伊藤貴之：力学モデルを用いた階層型グラフデータ画面配置手法の改良手法とウェブサイト視覚化への応用, 芸術科学会論文誌, Vol.3, No.4, pp.250–263(2004).
- [9] 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).
- [10] Brøns, M. Jakobsen, B. Niss, K. Bisgaard, A. and Voigt, K. L.: Streamline topology in the near wake of a circular cylinder at moderate reynolds numbers, *Journal of Fluid Mechanics*, Vol.584, pp.23–43 (2007).
- [11] 谷崎正明, 丸山貴志子, 嶋田茂：デフォルメマップ生成のための道路形状正規化モデルとそのシステム評価, 電子情報通信学会論文誌. A, Vol.87, No.1, pp.108–119(2004).

*1 livedoor NEWS (2015 年 01 月 9 日), <
<http://news.livedoor.com/article/detail/9657771/>> (参照
2019-09-17)