Julia 言語で FEM を書いてポテンシャル流れを解く

きゅーしす

2021 年 12 月 21 日

概要

Julia 言語は近年開発された言語で, 簡潔に表現されたコードで高速に実行できる特徴をもつ. これらの特徴からデータサイエンスや機械学習, 科学技術計算での利用が広がっている. この例として, 筆者が自作した 2 次元 Poisson 方程式の有限要素法 (Finite Element method, FEM) のコードを取り上げ, Julia 言語の特徴を実例を交えながら示す.

1 イントロダクション

Julia 言語は近年開発された言語で, 自由なライセンスで高速で簡潔に記述でき, 強力なマクロがあり, 汎用性を持ち, 統計や科学技術計算にも強い [1]. また, パッケージ管理も十分に整備されており, Python をはじめとしたユーザーの多い言語には及ばないものの, パッケージは充実している.

汎用性がありつつも科学技術計算に強いという特徴やパッケージ管理システムが整備されていることは数値解析に大いに役立つ. たとえば, 線形代数や複素数が簡潔な記法で記述でき, 従来の数値計算で使われている C/C++ や Fortran の泣き所であったファイルの入出力や結果のプロットまで一貫して Julia 言語のみで構築できる. 一方で, C/C++ や Fortran はパッケージ管理システムに乏しく, その上メンテナンス性が高いとはとても言えない[*1]. さらに, プロファイリングや並列計算も便利な記法があり, 高性能計算 (HPC) での利用も広がっている.

Julia 言語は利点も多数あるが, 新しいプログラミング言語であるので情報が少ない. 海外では利用例が増えているが, 日本での利用例が少ないのも課題である. 科学技術計算での利用はは尚更その傾向が強い. この記事で日本語の Julia 言語での科学技術計算の情報を少しでも提供できたら幸いである.

続いて構成を示す. はじめに, 今回解くポテンシャル流れについて概説する. 次に有限要素法での近似と離散化, アルゴリズムについて概説する. その後に Julia 言語の特徴と書いたプログラムの実装を示す. そして計算結果と考察をし, 記事全体のまとめを記す.

2 ポテンシャル流れとは

今回解く問題はポテンシャル流れと呼ばれる. ポテンシャル流れの「ポテンシャル」とは何者なのか, 支配方程式は何なのか, 文献 [2] を参考にその導出を行って概説する.

ポテンシャル流れの対象となるのは非圧縮完全流体と呼ばれるものである. 性質を以下に列挙する.

[*1] 個人の見解です. 異論は認めます

- **非圧縮**とは流体の密度が変わらないこと ($\partial\rho/\partial t = 0$)[*2]
- **完全流体**は非粘性である (粘性がない)
- 完全流体は非熱伝導性である (熱を一切伝えない)

なお, 完全流体の定義は神部ら [2] の定義を採用した. 非圧縮完全流体の支配方程式は以下に示す連続の式 (式 (1)) と Euler の運動方程式 (式 (2)) である.

$$\nabla \cdot \boldsymbol{u} = 0 \tag{1}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\frac{1}{\rho}\nabla p + \boldsymbol{f} \tag{2}$$

ここで速度を $\boldsymbol{u}$, 密度を ρ, 圧力を p, 外力を $\boldsymbol{f}$ とした. Euler 方程式は非圧縮かつ完全流体という仮定を入れたにもかかわらず, 解析的な解を求めることはほぼ不可能である[*3]. さらに幾つかの仮定を導入してより簡単な方程式を導出し, 流れを近似的に求めることを行いたい.

まず, 流れ場が時間に対して変化しない定常状態 ($\partial\boldsymbol{u}/\partial t = 0$) を仮定する. このとき, Euler 方程式は以下のようになる.

$$(\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\frac{1}{\rho}\nabla p + \boldsymbol{f} \tag{3}$$

次に, 渦度 $\boldsymbol{\omega}$ を速度の回転 $\nabla \cdot \boldsymbol{u}$ と定義する. このとき, 外力がポテンシャル χ によるもので $\boldsymbol{f} = -\nabla\chi$ と表されれば, 式 (3) は以下のように変形できる.

$$\boldsymbol{\omega} \times \boldsymbol{u} = -\nabla\left(\frac{1}{2}|\boldsymbol{u}|^2 + \frac{p}{\rho} + \chi\right) \tag{4}$$

この式の右辺は速度と渦度のそれぞれに直交したベクトルである. つまり, 式の値は流線 ($\boldsymbol{u}$ にそった線) と渦線 ($\boldsymbol{\omega}$ にそった線) それぞれの上で $\frac{1}{2}|\boldsymbol{u}|^2 + \frac{p}{\rho} + \chi$ が変化しないことを示している. よって, 渦線もしくは流線に沿って

$$\frac{1}{2}|\boldsymbol{u}|^2 + \frac{p}{\rho} + \chi = \text{constant}, \quad \text{渦線もしくは流線に沿って} \tag{5}$$

が成り立つ. これを Bernoulli の定理という. 詳細は神部ら [2] を参照されたい.

次に**渦なし**という概念を導入する. 渦なしとはいたるところで渦度が 0 であることを意味する. 渦なしの場合, Stokes の定理より

$$\oint_C \boldsymbol{u} \cdot \mathrm{d}\boldsymbol{l} = \iint_S \nabla \times \boldsymbol{u} \cdot \boldsymbol{n}\mathrm{d}S = 0 \tag{6}$$

が成立する, ここで C は単純閉曲線で, S は C を境界とする面である. この結果から, 積分路に依存せず, 開始と終点のみで値が決まる. 速度ポテンシャルが存在する[*4]. 速度ポテンシャル Φ は

$$\boldsymbol{u} = \nabla\Phi \tag{7}$$

[*2] 非圧縮な流体は音速が無限大になるので現実世界には存在しない
[*3] この問題は大変難しく, 未だ解析解があるかどうかすらわかっていない
[*4] 詳細は付録 A.1 を参照

を満たす．ポテンシャル流れのポテンシャルとはこの速度ポテンシャルのことを意味する．[*5]連続の式 (1) に式 (7) を代入すれば Laplace 方程式

$$\nabla^2 \Phi = 0 \tag{8}$$

が導けた．

ここで Bernoulli の定理を思い出して, 渦なしの場合はどうなるかを考察する．式 (4) は渦なしの場合,

$$\nabla \left(\frac{1}{2}|\boldsymbol{u}|^2 + \frac{p}{\rho} + \chi \right) = 0 \tag{9}$$

となり, すべての領域において

$$\frac{1}{2}|\boldsymbol{u}|^2 + \frac{p}{\rho} + \chi = \text{constant}, \quad \text{for all area} \tag{10}$$

これも Bernoulli の定理として知られている．

最後に境界条件について述べる．壁境界は壁に対して流体は浸透しないので, 壁境界の外向き法線ベクトルを $\boldsymbol{n}_{\text{wall}}$ とすると

$$\boldsymbol{u} \cdot \boldsymbol{n}_{\text{wall}} = \nabla \Phi \cdot \boldsymbol{n}_{\text{wall}} = 0 \tag{11}$$

となる．次に流体の流入境界を考える．流入境界は流体領域の外向き法線ベクトルを $\boldsymbol{n}_{\text{in}}$ として, 流入速度を U_{in} としたときに

$$\boldsymbol{u} = -U_{\text{in}} \boldsymbol{n}_{\text{in}} \tag{12}$$

$$\nabla \Phi \cdot \boldsymbol{n}_{\text{in}} = -U_{\text{in}} \boldsymbol{n}_{\text{in}} \cdot \boldsymbol{n}_{\text{in}} \tag{13}$$

$$\nabla \Phi \cdot \boldsymbol{n}_{\text{in}} = -U_{\text{in}} \tag{14}$$

が得られる．$\boldsymbol{n}_{\text{in}} \cdot \boldsymbol{n}_{\text{in}} = 1$ を用いた．最後に流出境界条件を考える．流出境界は外向き法線ベクトル $\boldsymbol{n}_{\text{out}}$ に沿って一様の速さで流体が流出するとモデリングする[*6]．このとき, 接線ベクトル $\boldsymbol{t}_{\text{out}}$ と速度場で以下の関係が成立する．

$$\boldsymbol{u} \cdot \boldsymbol{t}_{\text{out}} = \boldsymbol{\nabla} \Phi \cdot \boldsymbol{t}_{\text{out}} = 0 \tag{15}$$

つまり, 流出境界に沿って速度ポテンシャル Φ 一切の変化がない．よって速度ポテンシャルは流出境界において同じ値をとり, 流出境界は Dirichlet 条件となる[*7]．

3　有限要素解析

有限要素法とは領域を有限個の要素に分割して, その要素ごとに代数方程式を立式して領域全体の偏微分方程式を解く近似解法である．

[*5] 付録 A.1 を参照
[*6] これは計算しやすいため導入している．また流体の流出条件の与え方も様々な研究がある．
[*7] 一般にポテンシャルは基準のとり方によって変わるので境界条件の値は自由に決められる

文献 [3] を参考にして式 (8) の Laplace 方程式を解く．取り扱いの都合上, 流入・流出・壁境界を Dirichlet 境界条件と Neumann 境界条件にまとめて, 以下のように表記する．

$$\nabla^2 u(\boldsymbol{x}) = 0, \quad \text{in}\,\Omega \tag{16}$$
$$u(\boldsymbol{x}) = f_D, \quad \text{on}\,\partial\Omega_D \tag{17}$$
$$\nabla u(\boldsymbol{x}) \cdot \boldsymbol{n} = f_N, \quad \text{on}\,\partial\Omega_N \tag{18}$$

ここで, $\partial\Omega_D$ は Dirichlet 境界が貼られている境界で, $\partial\Omega_N$ が Neumann 境界条件が貼られている境界である．

次に式 (16) に対して, 重み付き残差を適用して弱形式を得る．残差 r を

$$r(\boldsymbol{x}) = \nabla^2 u(\boldsymbol{x}) \tag{19}$$

と定義する．残差に対して重み関数 $w(\boldsymbol{x})$ を乗じて計算領域で積分をする．

$$\int_\Omega w(\boldsymbol{x}) r(\boldsymbol{x})\,\mathrm{d}\Omega = 0 \tag{20}$$

式 (16) の両辺に重み関数を乗じて積分しているので当然右辺は 0 となる．この重み付き残差に含まれるラプラシアンが今後の計算で不都合があるため, Gauss-Green の定理[*8]

$$\int_\Omega \phi \nabla^2 \psi\,\mathrm{d}\Omega = -\int_\Omega \nabla\phi \cdot \nabla\psi\,\mathrm{d}\Omega + \int_{\partial\Omega} \phi \nabla\psi \cdot \boldsymbol{n}\,\mathrm{d}\Gamma \tag{21}$$

を用いて

$$-\int_\Omega \nabla w \cdot \nabla u\,\mathrm{d}\Omega + \int_{\partial\Omega} w \nabla u \cdot \boldsymbol{n}\,\mathrm{d}\Gamma = 0 \tag{22}$$
$$\int_\Omega \nabla w \cdot \nabla u\,\mathrm{d}\Omega = \int_{\partial\Omega_N} w f_N\,\mathrm{d}\Gamma \tag{23}$$

となり, 弱形式化された Laplace 方程式 (23) を得られた．最後の式変形では Neumann 条件 (式 (18)) を代入した．

次に Galerkin 法を適用する．Galerkin 法は弱形式化した偏微分方程式を解く方法で, 解を有限個の基底関数の重み付き和で近似し, 重み関数を解と同じ基底関数として解く．また, 同時に領域内部に有限個の接点を配置し, 領域を分割する．

$$u \approx u_h(\boldsymbol{x}) = \sum_{i=1}^{M} u_i N_i(\boldsymbol{x}) \tag{24}$$

ここで u_i は係数, N_i は基底関数である．ベクトルを用いて表記すると

$$u_h(\boldsymbol{x}) = \boldsymbol{u}_i \boldsymbol{N}_i(\boldsymbol{x}) \tag{25}$$

となる．基底関数は形状関数と呼ばれる．形状関数はコンパクトな関数で, 様々な形状を表現でき, Partition of Unity 条件を満たす．コンパクトであるので形状関数は図 1a のような形状をしており, 接点 j において

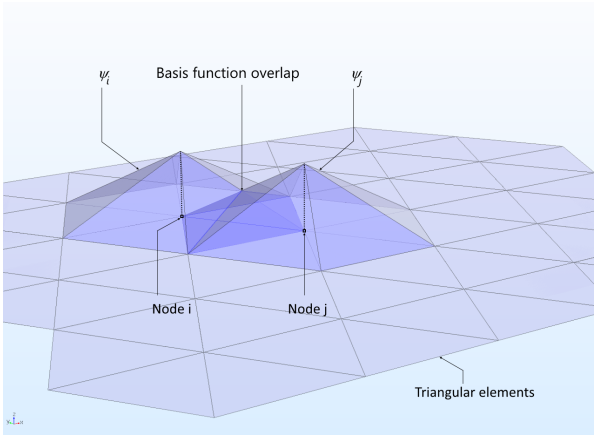
$$N_i(\boldsymbol{x}_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{26}$$

[*8] 導出については A.2 を参照

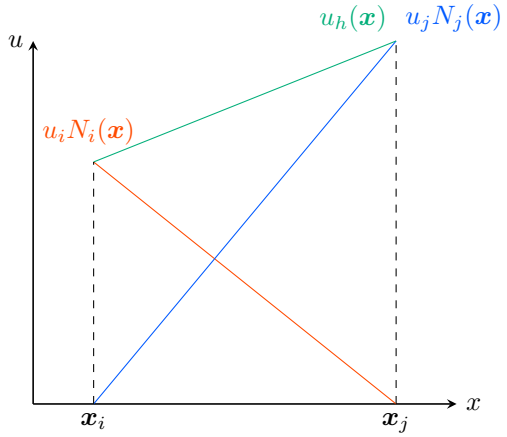
を満たす．Partition of Unity 条件とは以下の条件を満たすことである．

$$\sum_{i=1}^{M} N_i(\boldsymbol{x}) = 1, \quad \text{for all } \boldsymbol{x} \in \Omega \tag{27}$$

形状関数で u を表現している様子は図 1b を参照されたい．式 (23) に Galerkin 法を適用すると，式 (25) の定



(a) 2 つの接点にある形状関数 Web サイト [4] より引用



(b) 形状関数を用いた補間

図 1: 形状関数による補間

義より，u は定数だから積分の外に出せることに注意して

$$\sum_{j=1}^{M} \left[\int_{\Omega} \nabla N_i \cdot \nabla N_j \, \mathrm{d}\Omega \right] u_j = \int_{\partial\Omega_N} N_i f_N \, \mathrm{d}\Gamma \tag{28}$$

を得る．これは M 本の連立方程式でベクトル表記すると以下の式を得る．

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f} \tag{29}$$

$$A_{ij} = \int_{\Omega} \nabla N_i \cdot \nabla N_j \, \mathrm{d}\Omega \tag{30}$$

$$\boldsymbol{u} = \begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_M \end{Bmatrix} \tag{31}$$

$$f_i = \int_{\partial\Omega_N} N_i f_N \, \mathrm{d}\Gamma \tag{32}$$

式 (28) によると領域全体で積分を行い $\boldsymbol{u}$ を求められるが，領域全体で積分するのは実装が難しい．コードの実装では要素ごとに積分を行ってその後に足し合わせて解を求める．

ここに有限要素法の大きな利点がある．有限要素法は要素ごとに領域を分割し，要素内部で立式することができれば，どのような形状も扱うことができるという強みを持つ．また，分割する要素の形状も大きさも一定でなくてよく，異なる形状の要素を混在させることもできる．

領域を要素ごとに分割して代数方程式を立てる．今回は要素はすべて三角形 1 次要素であるとする．領域 Ω は三角形要素 Ω^e で分割される．したがって弱形式 (23) も要素 Ω^e 上で成り立つ．ここで要素内部での節点番号が全体での節点番号と異なるので, 右上に e をつけて区別する．

$$\int_{\Omega^e} \nabla N_i^e \cdot \nabla N_j^e u_j^e \,\mathrm{d}\Omega = \int_{\partial\Omega_N^e} N_i f_N \,\mathrm{d}\Gamma \tag{33}$$

境界条件の導入は全体マトリックスを得た後に行うため, Neumann 境界条件の積分は境界では場所だと相互に打ち消されるため, Neumann 境界条件は Neumann 境界のみになる．

また, Neumann 境界条件の導入のしやすさも有限要素法の利点である．差分法では Neumann 境界条件の導入は煩雑な処理が必要であるが, 有限要素法は差分法よりも簡単に導入できる．さらに, Neumann 境界条件が 0 の場合 (今回だと壁境界, 熱伝導では断熱, 固体力学では外力なし) は何も処理しなくて導入できるのは大きな利点である．

三角形一次要素は三角形の頂点 1, 2, 3, において

$$N_i^e(\boldsymbol{x}_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{34}$$

$$N_1^e + N_2^e + N_3^e = 1 \tag{35}$$

を満たす形状関数を持つ．形状関数 N_i は

$$N_i^e = a_i^e + b_i^e x + c_i^e y \tag{36}$$

と表される．係数はそれぞれ

$$\begin{bmatrix} 1 & x_1^e & y_1^e \\ 1 & x_2^e & y_2^e \\ 1 & x_3^e & y_3^e \end{bmatrix} \begin{bmatrix} a_1^e & a_2^e & a_3^e \\ b_1^e & b_2^e & b_3^e \\ c_1^e & c_2^e & c_3^e \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{37}$$

の関係を満たす．そこで係数について解くと,

$$a_i = \frac{x_j y_k - x_k y_j}{2|\Omega^e|}, \quad b_i = \frac{y_j - y_k}{2|\Omega^e|}, \quad c_i = \frac{x_k - x_j}{2|\Omega^e|} \tag{38}$$

と表される．ここで i, j, k は 1,2,3 の偶置換[*9] であり, $|\Omega^e|$ は要素の面積である．

次に式 (33) の弱形式を行列形式にまとめるとこのようになる．

$$\boldsymbol{A}^e \boldsymbol{u}^e = \boldsymbol{f}^e \tag{39}$$

ここで $\boldsymbol{A}^e$ は要素剛性マトリックスと呼ばれ, 以下の式となる．

$$\boldsymbol{A}_{ij}^e = \int_{\Omega^e} \left(\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} \right) \mathrm{d}V \tag{40}$$

式 (40) は形状関数 (式 (36)) を代入して

$$\begin{aligned} \boldsymbol{A}_{ij}^e &= \int_{\Omega^e} \left(\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} \right) \mathrm{d}V \\ &= \int_{\Omega^e} (b_i b_j + c_i c_j) \,\mathrm{d}V \\ &= (b_i b_j + c_i c_j)|\Omega^e| \end{aligned} \tag{41}$$

[*9] $((i,j,k) = ((1,2,3),(2,3,1),(3,1,2)))$

となる．ここで $|\Omega^e|$ は領域の面積である．$\boldsymbol{f}^e$ は等価接点流量とよばれる．

$$\boldsymbol{f}^e = \int_{\partial\Omega_N^e} \boldsymbol{N}^e f_N \,\mathrm{d}\Gamma \tag{42}$$

この積分は計算を解析的に求めることは難しいので, 近似計算で求める．はじめに要素 e の境界の積分路を図 2 のように定める．ここで Γ_1^e の境界積分を考える．経路 Γ_1^e は $\boldsymbol{x}_2^e$ から $\boldsymbol{x}_3^e$ に向かう経路であり, その経路は

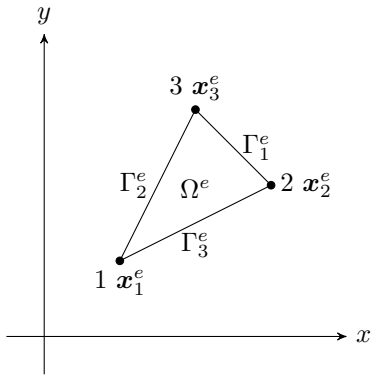


図 2: 境界の積分路

パラメータ t を用いて表すと $\boldsymbol{\gamma}(t) = \frac{(1+t)}{2}\boldsymbol{x}_3 + \frac{(1-t)}{2}\boldsymbol{x}_2$ となる．Γ_1 の境界積分は線積分となるので

$$\begin{aligned}\int_{\Gamma_1^e} \boldsymbol{N}^e f_N \,\mathrm{d}\Gamma &= \int_{-1}^{1} \boldsymbol{N}^e(\boldsymbol{\gamma}(t)) f_N(\boldsymbol{\gamma}(t)) |\boldsymbol{\gamma}(t)| \,\mathrm{d}t \\ &= \int_{-1}^{1} \boldsymbol{N}^e(\boldsymbol{\gamma}(t)) f_N(\boldsymbol{\gamma}(t)) \frac{|\boldsymbol{x}_3^e - \boldsymbol{x}_2^e|}{2} \,\mathrm{d}t\end{aligned} \tag{43}$$

を得る．次に Gauss 求積という手法を用いて先程の積分を近似する．Gauss 求積は被積分関数 ϕ に対して以下の式で表される．

$$\int_{-1}^{1} \phi(x) \,\mathrm{d}x \approx \sum_{i=1}^{n} w_i \phi(x_i) \tag{44}$$

有限個の積分点 x_i に対して重み w_i を乗じてその和で近似する．今回は積分点 2 点で計算する．このとき積分点は $\pm\frac{1}{\sqrt{3}}$, 重みは 1 となることが知られている [5].

他の積分路でも同様の計算が成立するので境界積分が打ち消されない場合, 式 (42) は i, j, k は $1, 2, 3$ の偶置換として, 以下のように積分を近似できる．

$$\boldsymbol{\gamma}_i(t) = \frac{1+t}{2}\boldsymbol{x}_k^e + \frac{1-t}{2}\boldsymbol{x}_j^e \tag{45}$$

$$\begin{aligned}\boldsymbol{f}^e &= \int_{\partial\Omega_N^e} \boldsymbol{N}^e f_N \,\mathrm{d}\Gamma \\ &= \sum_{i=1}^{3} \left[\int_{-1}^{1} \boldsymbol{N}^e(\boldsymbol{\gamma}_i(t)) f_N(\boldsymbol{\gamma}_i(t)) \frac{|\boldsymbol{x}_k^e - \boldsymbol{x}_j^e|}{2} \,\mathrm{d}t \right] \\ &\approx \sum_{i=1}^{3} \left[\sum_{l=1}^{2} \boldsymbol{N}^e\left(\boldsymbol{\gamma}_i\left(\frac{(-1)^2}{\sqrt{3}}\right)\right) f_N\left(\boldsymbol{\gamma}_i\left(\frac{(-1)^2}{\sqrt{3}}\right)\right) \frac{|\boldsymbol{x}_k^e - \boldsymbol{x}_j^e|}{2} \right]\end{aligned} \tag{46}$$

さらに先程計算した要素剛性マトリックスから全体剛性マトリックスを求める．要素剛性マトリックスから全体剛性マトリックスを求める操作はアセンブル操作と呼ばれる．アセンブル操作の模式図を図 3 に示す．具

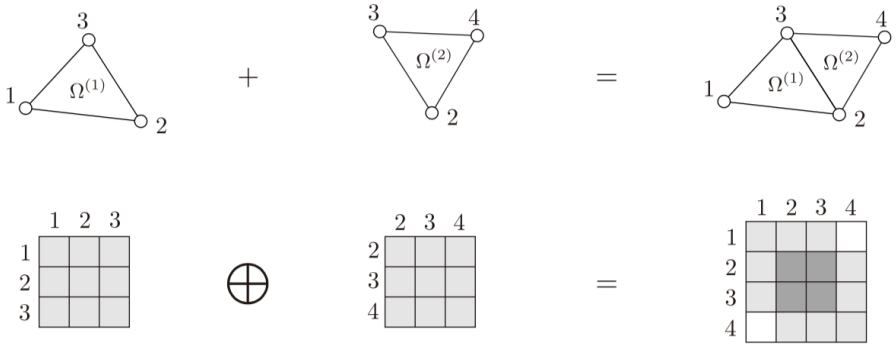


図 3: 領域ごとのアセンブル操作の概略 Web サイト [6] より引用

体的なアセンブルのアルゴリズムをアルゴリズム 1 に示す．nodes は接点で M 個の接点座標の配列で，faces は要素を指定するノードが並んでいいる 2 次元配列で，一つの行に要素の nodes に対するインデックスが 3 つ並んでいる．ただし，上付き文字の文字は配列を意味しない．また，A は疎行列となるので，多くの場合は A を

Algorithm 1 Assembling global stiffness matrix

Input: nodes, faces
Output: A

```
A ← O
for e ← 1, 2, ..., N do
    for i ← 1, 2, 3 do
        x^e[i] ← node[faces[e, i]]
    end for
    calculate b, c using eq. (38)
    for i, j ← 1, 2, 3 do
        A^e[i, j] ← (b[i]b[j] + c[i]c[j])|Ω^e|
    end for
    for i, j ← 1, 2, 3 do
        A[faces[e, i], faces[e, j]] ← A[faces[e, i], faces[e, j]] + A^e[i, j]
    end for
end for
```

CSR (Compressed Sparse Row) 形式や CSC (Compressed Sparse Column) 形式で格納される．

なお，局所等価接点流量ベクトルも同様に全体等価接点流量ベクトルをアセンブル操作ができる．

最後に Dirichlet 条件の代入を行う．接点 i が Dirichlet 条件 u_D だった場合を述べる．連立方程式を解いたときもその条件が維持される必要があるので i 行目は $u_i = u_D$ が成立して，他の行に u_i の項が含まれないようにする．また，全体剛性行列は対象行列であるという性質が数値計算の上では便利であるため，対称性を残す．これら 2 つの要件をアルゴリズムにするとアルゴリズム 2 が得られる．疎行列のデータ形式は 0 の領域を

Algorithm 2 Applying Dirichlet boundary condition

Input: $A \in \mathbb{R}^{M\times M}$, $f \in \mathbb{R}^M$, i, u_D
Output: A, f
  $A[i,i] \leftarrow 1$
  $f[i] \leftarrow u_D$
  for $j \neq i$ **do**
    $A[i,j] \leftarrow 0$
    $f[j] \leftarrow f[j] - A[j,i]u_D$
    $A[j,i] \leftarrow 0$
  end for

保存せずに圧縮するので 0 の入った項を最後に削除するコードを実装するとよい.

4 翼型の生成

今回は計算対象として翼型のまわりの流れを計算する. 翼型はさまざまなものがあるが, NACA 翼型の一つである NACA 2412 を用いる. NACA 翼型は NACA(National Advisory Committee for Aeronautics, 連邦航空諮問委員会, NASA の前身) が開発した翼型である.

文献 [7] を参考に NACA 2412 を説明する. NACA 2412 は NACA 4 桁翼型と呼ばれるグループに属しており, 翼型断面が多項式で定義されている. 翼型の厚み分布 y_t, 前縁半径 r_t, 翼型中心線 y_c を用いて定義する.

$$y_t = 5t\left(0.2969\sqrt{x} - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4\right) \tag{47}$$

$$r_t = 1.1019t^2 \tag{48}$$

$$y_c = \begin{cases} \frac{m}{p^2}(2px - x^2), & 0 \leq x \leq p \\ \frac{m}{(1-p)^2}[(1-2p) + 2px - x^2], & p < x \leq 1 \end{cases} \tag{49}$$

また, パラメータとして

- x : 翼弦の相対的な位置 (前縁を 0 とし, 後縁を 1 とする)
- m : 翼型の厚みの比 (翼型の番号の 1000 の位を 100 で割る, 今回は $m = 0.02$)
- p : 反りが最大になる x の値 (翼型の番号の 100 の位を 10 で割る, 今回は $p = 0.4$)
- t : 翼型の最大厚みの比 (翼型の番号の下二桁を 100 で割る, 今回は $t = 0.12$)

がある. 主翼の上半分を構成する曲線を (x_U, y_U) とし, 下半分を構成する曲線を (x_L, y_L) とする. このとき,

$$x_U = x - y_t \sin\theta, \qquad y_U = y_c + y_t \cos\theta \tag{50}$$

$$x_L = x + y_t \sin\theta, \qquad y_L = y_c - y_t \cos\theta \tag{51}$$

と定義される [8]. ここで θ は翼型中心線の接線と x 軸のなす角で, 以下の式で与えられる.

$$\theta = \arctan\frac{\mathrm{d}y_c}{\mathrm{d}x} \tag{52}$$

Julia 言語で Zhukovsky 変換を実装したコードをリスト 1 に示し[*10], 画像を図 4 に示す.

Listing 1 NACA2412 の計算とプロットのコード

```
using Plots

#NACA 2412 のパラメータ
m = 0.02
p = 0.4
t = 0.12

yₜ(x) = 5t * (0.2969 * sqrt(x) - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4)
y_c(x) = ifelse(x <= p, m / (p^2) * (2p * x - x^2), m / ((1 - p)^2) * ((1 - 2p) + 2p * x - x^2))
dy_c_dx(x) = ifelse(x <= p, m / (p^2) * (p - x), 2m / ((1 - p)^2) * (p - x))
θ(x) = atan(dy_c_dx(x))

xᵤ = []; yᵤ = []; xₗ = []; yₗ = []
# multi resoluton range
# https://discourse.julialang.org/t/combine-multiple-ranges-with-different-resolution/36822/7
for x in sort(unique(vcat(range(0, 0.1, length = 20), range(0.1, 1, length = 20))))
    push!(xᵤ, x - yₜ(x) * sin(θ(x)))
    push!(yᵤ, y_c(x) + yₜ(x) * cos(θ(x)))
    push!(xₗ, x + yₜ(x) * sin(θ(x)))
    push!(yₗ, y_c(x) - yₜ(x) * cos(θ(x)))
end
x_foil = vcat(xᵤ, reverse(xₗ))
y_foil = vcat(yᵤ, reverse(yₗ))

x_mean = [x for x in sort(unique(vcat(range(0, 0.1, length = 20), range(0.1, 1, length = 20))))]
y_mean = y_c.(x_mean)

plot(x_mean, y_mean, label = "\$y_c\$")
plot!(x_foil, y_foil, label = "NACA2412")
plot!(; aspect_ratio = 1.0, legendfontsize = 16,
    axis = nothing, showaxis = false, grid = false)
savefig("2d_velocity_potential/NACA2412.pdf")
```

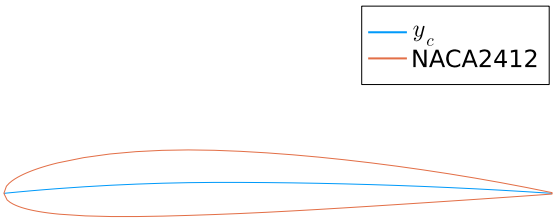


図 4: NACA 2412 翼型

このように, 数式とコードが近く, 使える文字が多いのが Julia 言語の特徴である. たとえば関数を数式のように定義したり (8 行から 12 行), ベクトルの配列を柔軟に追加できたり, べき乗の計算も簡単に行えたりとコードの表現を簡潔にできるまた, プロットライブラリも充実しており, 計算コードとプロットを一つにまとめられるという利点がある[*11]

次にこの翼型を解析領域でメッシュを切って要素ごとに分割を行う. メッシャーとして Gmsh[9] とそのラッパーである Gmsh.jl[*12]を用いる.

5 結果と考察

計算結果が解析解と定性的に一致しており, 正しく問題を解くことができたと考えられる.

[*10] 下付き文字の l と t はフォントに対応する文字がなかったので小さい l と t でごまかしている
[*11] 筆者は C++ と Python で計算とプロットが分かれているものを Julia で統合しつつある.
[*12] https://github.com/JuliaFEM/Gmsh.jl

参考文献

[1] Jeff Bezanson et al. *Why We Created Julia*. 2012. URL: https://julialang.org/blog/2012/02/why-we-created-julia/.

[2] 神部 勉 and 石井 克哉. **流体力学**. 第 8 版. 裳華房, 1995. ISBN: 978-4785320638.

[3] Mats G. Larson and Fredrik Bengzon. *The Finite Element Method: Theory, Implementation, and Applications*. Springer Berlin Heidelberg, 2013. DOI: 10.1007/978-3-642-33287-6.

[4] COMSOL inc, ed. *Detailed Explanation of the Finite Element Method (FEM)*. 2017. URL: https://www.comsol.jp/multiphysics/finite-element-method.

[5] 安田 仁彦. **数値解析基礎**. コロナ社, 2008. ISBN: 978-4339060973.

[6] nqomorita. **有限要素法と反復法線形ソルバのはじめの一歩** *[前編]*. 2018. URL: https://qiita.com/nqomorita/items/6fc4940c1a9532f02c10.

[7] 李家 賢一. **航空機設計法-軽飛行機から超音速旅客機の概念設計まで-**. コロナ社, 2011. ISBN: 978-4339046199.

[8] Ira H Abbott, Albert E Von Doenhoff, and Louis Stivers Jr. *Summary of airfoil data*. Tech. rep. National Advisory Committee for Aeronautics, 1945.

[9] Christophe Geuzaine and Jean-François Remacle. "Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities". In: *International journal for numerical methods in engineering* 79.11 (2009), pp. 1309–1331.

付録 A　諸々の計算

A.1　速度ポテンシャルの存在

以下の式で定義される関数 Φ が速度ポテンシャルであることを示す.

$$\Phi(\boldsymbol{x}) = \int_O^{\boldsymbol{x}} \boldsymbol{u} \cdot \mathrm{d}\boldsymbol{l} \tag{53}$$

非圧縮かつ定常でその上渦なし完全流体が流れている場を考察する. 場の内部に 2 点 P, Q があり, P から出発して Q で終わる異なる曲線 C_1, C_2 があり, C_1 と C_2 で囲まれる領域が曲面となっているものとする. 点 P から C_1 を通って点 Q にいたり, 点 Q から C_2 を通って点 P に戻る積分路を考える. この周回積分の値は式 (6) より 0 となることが示されるので

$$\oint_{C_1-C_2} \boldsymbol{u} \cdot \mathrm{d}\boldsymbol{l} = 0 \tag{54}$$

$$\int_{C_1} \boldsymbol{u} \cdot \mathrm{d}\boldsymbol{l} - \int_{C_2} \boldsymbol{u} \cdot \mathrm{d}\boldsymbol{l} = 0 \tag{55}$$

$$\int_{C_1} \boldsymbol{u} \cdot \mathrm{d}\boldsymbol{l} = \int_{C_2} \boldsymbol{u} \cdot \mathrm{d}\boldsymbol{l} \tag{56}$$

以上の式変形から, 式 (53) で定めた関数 Φ の値は積分路に依存しない. よって関数 Φ がポテンシャルである.

A.2 Gauss-Green の定理

Gauss の発散定理

$$\int_\Omega \nabla \cdot \boldsymbol{A}\,\mathrm{d}\Omega = \int_{\partial\Omega} \boldsymbol{A}\cdot\boldsymbol{n}\,\mathrm{d}\Gamma \tag{57}$$

に対して $\boldsymbol{A} = \phi\nabla\psi$ を代入し, 式変形をして Gauss-Green の定理を得る.

$$\int_\Omega \nabla\cdot(\phi\nabla\psi)\,\mathrm{d}\Omega = \int_{\partial\Omega}\phi\nabla\psi\cdot\boldsymbol{n}\,\mathrm{d}\Gamma \tag{58}$$

$$\int_\Omega (\phi\nabla^2\psi + \nabla\phi\cdot\nabla\psi)\,\mathrm{d}\Omega = \int_{\partial\Omega}\phi\nabla\psi\cdot\boldsymbol{n}\,\mathrm{d}\Gamma \tag{59}$$

$$\int_\Omega \phi\nabla^2\psi\,\mathrm{d}\Omega = -\int_\Omega \nabla\phi\cdot\nabla\psi\,\mathrm{d}\Omega + \int_{\partial\Omega}\phi\nabla\psi\cdot\boldsymbol{n}\,\mathrm{d}\Gamma \tag{60}$$

付録 B ボツになった内容

B.1 Zhukovsky 変換を用いた翼の生成

Zhukovsky 変換を用いて翼型を生成した. 現在の旅客機ではスーパークリティカル翼型をベースに設計されているが [7][*13], 簡単な変数変換によって生成でき, 2 次元翼理論による結果の検証がしやすいためこれを用いる.

Zhukovsky 変換は複素平面 ζ から複素平面 z に対して以下の複素関数で変換を行うものである.

$$z = \zeta + \frac{1}{\zeta} \tag{61}$$

この変換の特徴として, z を円とすると z が翼型になるということと, $\zeta \neq 0$ において正則関数になる特徴がある. 今回は ζ を以下のように定めた.

$$|\zeta - \zeta_0| = |c - \zeta_0| \tag{62}$$

ここで, $\zeta_0 = -0.1 + 0.1i$, $c = 1$ と定めた.

Julia 言語で Zhukovsky 変換を実装したコードをリストに示し, 画像を図 5 に示す.

[*13] 2 次元翼型については文献 [7] の 5 章が詳しい

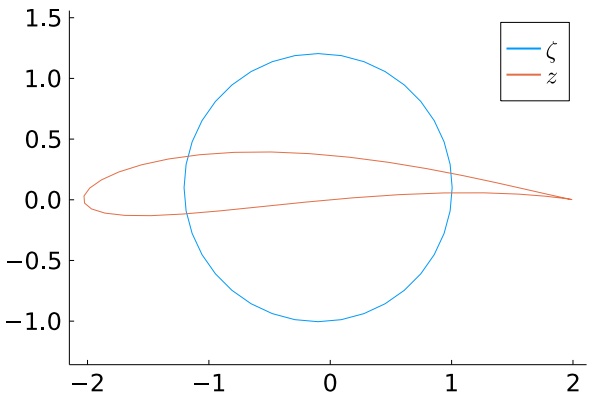


図 5: Zhukovsky 変換の結果