

# 一般流束条件での 定常 Navier-Stokes 方程式の数値解析

明治大学大学院  
理工学研究科 基礎理工学専攻 数学系  
福嶋 剛史  
指導教員 森本 浩子教授

2005年2月25日

# 目次

<b>1</b>	<b>初めに</b>	<b>2</b>
1.1	初めに . . . . .	2
<b>2</b>	<b>準備</b>	<b>5</b>
2.1	記号と関数空間の説明 . . . . .	5
<b>3</b>	<b>定常 Stokes 方程式</b>	<b>6</b>
3.1	問題設定と弱定式化 . . . . .	6
3.2	離散化したときの弱形式 . . . . .	7
3.3	要素係数行列の計算 . . . . .	11
3.4	近似方程式の作成 . . . . .	19
<b>4</b>	<b>定常 Navier-Stokes 方程式</b>	<b>21</b>
4.1	非線形項の弱定式化 . . . . .	21
4.2	Newton 法 . . . . .	25
<b>5</b>	<b>数値実験</b>	<b>29</b>
5.1	実験の問題設定 . . . . .	29
5.1.1	数値実験 1 . . . . .	29
5.1.2	数値実験 2 . . . . .	31
5.2	実験結果 . . . . .	31
5.2.1	数値実験 1 の結果 . . . . .	31
5.3	数値実験 2 の結果 . . . . .	40
5.4	結果のまとめ . . . . .	57
<b>6</b>	<b>実験で用いたプログラム</b>	<b>57</b>
6.1	領域分割と境界値作成を行うプログラム . . . . .	57
6.2	Stokes 方程式を解くプログラム . . . . .	65
6.3	行列 $\kappa_{ijk}^x, \kappa_{ijk}^y$ の計算プログラム (Mathematica による) . . . . .	74
6.4	Navier-Stokes 方程式を解くプログラム . . . . .	74
6.5	流速のベクトル場を描くプログラム . . . . .	85
6.6	等圧線を描くプログラム . . . . .	90

# 1 初めに

## 1.1 初めに

本論文では以下のような定常 Navier-Stokes 方程式を考える.

$$\begin{cases} -\nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \frac{1}{\rho}\nabla p = \mathbf{f} & \text{in } \Omega, \\ -\operatorname{div}\mathbf{u} = 0 & \text{in } \Omega, \\ \mathbf{u} = \mathbf{b} & \text{on } \partial\Omega. \end{cases}$$

$\Omega$  は流体の占める 2 次元有界領域,  $\partial\Omega$  はその境界,  $\rho$  は流体の密度,  $\nu > 0$  は粘性係数,  $\mathbf{u} = (u, v)^T$  は未知の 2 次元速度ベクトル,  $p$  は未知の圧力,  $\mathbf{f} = (f, g)^T$  は既知の滑らかな関数で単位質量あたりに働く外力,  $\mathbf{b}(x, y)$  は Dirichlet 境界条件に対応する与えられた関数とする.

ただし,  $\mathbf{b}(x, y)$  は Gauss の発散定理

$$\iint_{\Omega} \operatorname{div}\mathbf{u} \, dx \, dy = \int_{\partial\Omega} \mathbf{b}\cdot\mathbf{n} \, dS = 0,$$

により,

$$\int_{\partial\Omega} \mathbf{b}\cdot\mathbf{n} \, dS = 0,$$

をみたさなければならない.

ここで,  $\mathbf{n}$  は  $\partial\Omega$  の外向き単位法線ベクトル,  $\int_{\partial\Omega} dS$  は  $\partial\Omega$  における境界積分を表す記号とする.

また, 境界  $\partial\Omega$  の連結成分が複数個, すなわち

$$\partial\Omega = \Gamma_1 \cup \Gamma_2 \cup \dots \cup \Gamma_N \quad (\Gamma_i \text{ は連結成分}),$$

の場合, 上の条件は以下のように書き表せる.

$$\int_{\partial\Omega} \mathbf{b}\cdot\mathbf{n} \, dS = \sum_{i=1}^N \int_{\Gamma_i} \mathbf{b}\cdot\mathbf{n} \, dS = 0.$$

この条件を一般流束条件という.

各  $i$  ( $i = 1, 2, \dots, N$ ) について

$$\int_{\Gamma_i} \mathbf{b}\cdot\mathbf{n} \, dS = 0.$$

をみたすならば, 一般流束条件をみたす. このときは Navier-Stokes 方程式の解の存在が解析的に示されている. (H.Fujita, O.A.Ladyzhenskaya)

しかし, この条件が成立せず, 一般流束条件のみが成立する場合は, 解が一般的にあるかどうかは知られていない.

そこで, 本論文では一般流束条件のみが成立するような問題を考え, その問題に対して 2 つの数値実験を行ったので紹介する.

領域  $\Omega$  を以下のような円環領域に設定する.

$$\Omega = \{\mathbf{x} \in \mathbb{R}^2; a < |\mathbf{x}| < 1\}$$

ただし  $a$  は  $0 < a < 1$  をみたす定数とする. また, 境界  $\Gamma_1, \Gamma_2$  を次のようにおく.

$$\Gamma_1 = \{\mathbf{x} \in \mathbb{R}^2; |\mathbf{x}| = a\}$$

$$\Gamma_2 = \{\mathbf{x} \in \mathbb{R}^2; |\mathbf{x}| = 1\}$$

$\mathbf{e}_r, \mathbf{e}_\theta$  を動径方向, 偏角方向の単位ベクトルとして,  $\mu, \omega_1, \omega_2$  は与えられた定数,  $R_1 = a, R_2 = 1$  とする.

Navier-Stokes 方程式の境界値  $\mathbf{b}$  を

$$\mathbf{b} = \frac{\mu}{R_i} \mathbf{e}_r + R_i \omega_i \mathbf{e}_\theta \quad \text{on } \Gamma_i \quad (\mu \neq 0, i = 1, 2),$$

とすると,  $\mathbf{b}$  は一般流束条件のみをみたす.

なぜなら, 外向き法線ベクトル  $\mathbf{n}$  が

$$\mathbf{n} = \begin{cases} -\mathbf{e}_r & \text{on } \Gamma_1 \\ \mathbf{e}_r & \text{on } \Gamma_2 \end{cases}$$

なので,

$$\begin{aligned} \int_{\Gamma_1} \mathbf{b} \cdot \mathbf{n} dS &= \int_{\Gamma_1} \left( \frac{\mu}{a} \mathbf{e}_r + a \omega_1 \mathbf{e}_\theta \right) \cdot (-\mathbf{e}_r) dS \\ &= - \int_{\Gamma_1} \frac{\mu}{a} dS = - \int_0^{2\pi} \frac{\mu}{a} a d\theta \\ &= -2\pi\mu, \end{aligned}$$

$$\begin{aligned} \int_{\Gamma_2} \mathbf{b} \cdot \mathbf{n} dS &= \int_{\Gamma_2} (\mu \mathbf{e}_r + \omega_2 \mathbf{e}_\theta) \cdot \mathbf{e}_r dS \\ &= \int_{\Gamma_2} \mu dS = \int_0^{2\pi} \mu d\theta \\ &= 2\pi\mu. \end{aligned}$$

よって,  $\mu \neq 0$  ならば, 一般流束条件のみをみたす.

上の  $\mathbf{b}$  に対して,  $\nu = \rho = 1, \mathbf{f} = \mathbf{0}$  である場合は以下のような厳密解  $u_0$  が存在することが H.Morimoto[7] で示されている.

$$u_0 = \frac{\mu}{r} \mathbf{e}_r + b(\mu, r) \mathbf{e}_\theta$$

ただし,  $r$  は各点の原点からの距離で,  $b(\mu, r)$  は  $\mu$  の値によって以下のように変わる.

(1)  $\mu \neq -2\nu$  のとき,

$$b(\mu, r) = \frac{c_1}{r} + c_2 r^{1+\frac{\mu}{\nu}},$$

$$c_1 = \frac{\omega_1 a^2 - \omega_2 a^{2+\frac{\mu}{\nu}}}{1 - a^{2+\frac{\mu}{\nu}}}, \quad c_2 = \frac{\omega_2 - \omega_1 a^2}{1 - a^{2+\frac{\mu}{\nu}}}.$$

(2)  $\mu = -2\nu$  のとき,

$$b(\mu, r) = \frac{1}{r}(c_1 + c_2 \log r),$$

$$c_1 = \omega_2, \quad c_2 = \frac{\omega_1 a^2 - \omega_2}{\log a}.$$

1 つ目の数値実験としては, 数値計算で求めた Navier-Stokes 方程式の解がどの程度厳密解を近似しているかを確認する. そのために, 有限要素法における要素分割数と, 数値解と厳密解との誤差の相関関係を可視化してみる.

今度は, 上の境界値に摂動を加えた場合を考える.

$$\mathbf{b} = \left( \frac{\mu}{R_i} + \varphi_i(\theta) \right) \mathbf{e}_r + (R_i \omega_i + \psi_i(\theta)) \mathbf{e}_\theta \quad \text{on } \Gamma_i, \quad (i = 1, 2)$$

ただし,  $\varphi_i(\theta)$ ,  $\psi_i(\theta)$  は周期  $2\pi$  の滑らかな関数で, 以下の条件をみたす.

$$\int_0^{2\pi} \varphi_i(\theta) d\theta = 0, \quad \int_0^{2\pi} \psi_i(\theta) d\theta = 0$$

摂動を加えた場合も同様に  $\mu \neq 0$  のときは, 一般流束条件のみをみたす.

このとき,  $\nu, a, \omega_1, \omega_2$  が以下の条件

$$|\omega_1 - \omega_2| \frac{a^2}{1 - a^2} (\log a)^2 < 2\nu$$

をみたしている場合, ある 1 次元可算集合  $M$  に対し, 任意の  $\mu \in \mathbb{R} \setminus M$  として,  $\varphi_i, \psi_i$  ( $i = 1, 2$ ) が十分小さいならば Navier-Stokes 方程式の解が存在することが H.Morimoto and S.Ukai[8] で示されている.

2 つ目の数値実験は, このことを確認するとともに, この条件を満たさない場合でも解が存在するかどうかについて調べることである. 本論文では,

$$\nu = \rho = 1, \quad a = 0.5, \quad \mathbf{f} = 0,$$

$$(\varphi_i(\theta), \psi_i(\theta)) = (AR_i \cos \theta, BR_i \sin \theta) \quad (A, B \text{ は任意の実数}).$$

と固定して考える. このとき条件式を  $\omega_1, \omega_2$  について整理すると

$$|\omega_1 - \omega_2| < \frac{6}{(\log 2)^2} \doteq 12,488.$$

となるので, パラメータ  $\mu, \omega_1, \omega_2, A, B$  を色々変えた場合の解について調べる.

この論文の概略は以下のとおりである. まず, 第 2 節で準備として記号と関数空間を設定し, 第 3 節で定常 Stokes 方程式を 3 角形有限要素を用いた有限要素法により数値的に解く. 第 4 節では Stokes 方程式の数値解を初期値として, Newton 法を用いることで Navier-Stokes 方程式の近似解を求める. 第 5 節では, 数値実験の問題設定と実験結果を紹介する. 第 6 節では, 実験で用いたプログラムを紹介する.

## 2 準備

### 2.1 記号と関数空間の説明

本節では記号と関数空間の説明を行う。

考える領域  $\Omega$  は流体の占める 2 次元有界領域,  $\partial\Omega$  はその境界とする.  
本論文では,

$$\begin{aligned}\Omega &= \{\mathbf{x} \in \mathbb{R}^2; a < |\mathbf{x}| < 1\} \\ \Gamma_1 &= \{\mathbf{x} \in \mathbb{R}^2; |\mathbf{x}| = a\}, \quad \Gamma_2 = \{\mathbf{x} \in \mathbb{R}^2; |\mathbf{x}| = 1\} \\ \partial\Omega &= \Gamma_1 \cup \Gamma_2\end{aligned}$$

とする. ただし  $a$  は  $0 < a < 1$  をみたす定数.

また,  $R_1 = a$ ,  $R_2 = 1$  とおく.

$\rho$  は流体の密度とする.

$\nu > 0$  は粘性係数とする.

$\mathbf{u} = (u, v)^T$  は未知の 2 次元速度ベクトルとする.

$p$  は未知の圧力とする.

$\mathbf{f} = (f, g)^T$  は既知の滑らかな関数で単位質量あたりに働く外力とする.

$\mathbf{b}(x, y)$  は Dirichlet 境界条件に対応する与えられた関数とする.

本論文では, 摂動を考慮しない場合は,

$$\mathbf{b} = \frac{\mu}{R_i} \mathbf{e}_r + R_i \omega_i \mathbf{e}_\theta \quad \text{on } \Gamma_i, \quad (i = 1, 2)$$

摂動を考慮する場合は,

$$\mathbf{b} = \left( \frac{\mu}{R_i} + A \cos \theta \right) \mathbf{e}_r + (R_i \omega_i + B \sin \theta) \mathbf{e}_\theta \quad \text{on } \Gamma_i, \quad (i = 1, 2)$$

とする. ただし  $\mu$ ,  $\omega_1$ ,  $\omega_2$ ,  $A$ ,  $B$  は与えられた定数,  $\mathbf{e}_r$ ,  $\mathbf{e}_\theta$  は動径方向, 偏角方向の単位ベクトルとする.

関数空間  $L^2(\Omega)$ ,  $H^1(\Omega)$  を以下の式で定義する.

$$\begin{aligned}L^2(\Omega) &= \left\{ u : \Omega \rightarrow \mathbb{R}; \iint_{\Omega} |u|^2 dx dy < \infty \right\} \\ H^1(\Omega) &= \left\{ u \in L^2(\Omega); \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \in L^2(\Omega) \right\}\end{aligned}$$

### 3 定常 Stokes 方程式

#### 3.1 問題設定と弱定式化

Navier-Stokes 方程式の近似解を求めるためには、まず Stokes 方程式の近似解を求める必要がある。

領域  $\Omega$  を三角形分割したのち、Rits-Galerkin 法により近似方程式を作成し、それを解いて有限要素解を求める。考える Stokes 方程式は以下のようなものである。

$$\begin{aligned} \nu \Delta u(x, y) - \frac{1}{\rho} \frac{\partial p}{\partial x}(x, y) + f &= 0 && \text{in } \Omega, \\ \nu \Delta v(x, y) - \frac{1}{\rho} \frac{\partial p}{\partial y}(x, y) + g &= 0 && \text{in } \Omega, \\ -\operatorname{div} \mathbf{u} &= 0 && \text{in } \Omega, \\ \mathbf{u}(x, y) &= \mathbf{b}(x, y) && \text{on } \partial\Omega. \end{aligned}$$

次に、この問題に対する弱形式を考える。 $u^*, v^*$  は  $\partial\Omega$  上で 0 をみたす区分的双 2 次関数、 $p^*$  は区分的双 1 次関数とする。

上の第 1 式から第 3 式の両辺にそれぞれ  $u^*, v^*, p^*$  を掛けて  $\Omega$  上積分すると以下の式を得る。

$$\begin{aligned} \nu \iint_{\Omega} (\Delta u) u^* dx dy - \frac{1}{\rho} \iint_{\Omega} \frac{\partial p}{\partial x} u^* dx dy + \iint_{\Omega} f u^* dx dy &= 0, \\ \nu \iint_{\Omega} (\Delta v) v^* dx dy - \frac{1}{\rho} \iint_{\Omega} \frac{\partial p}{\partial y} v^* dx dy + \iint_{\Omega} g v^* dx dy &= 0, \\ - \iint_{\Omega} \frac{\partial u}{\partial x} p^* dx dy - \iint_{\Omega} \frac{\partial v}{\partial y} p^* dx dy &= 0 \end{aligned}$$

Green の公式, Gauss の公式,  $u^*, v^*$  が境界で 0 になることより, 上 2 式を次のように変形できる。

$$\begin{aligned} -\nu \iint_{\Omega} (\nabla u \cdot \nabla u^*) dx dy + \frac{1}{\rho} \iint_{\Omega} \frac{\partial u^*}{\partial x} p dx dy + \iint_{\Omega} f u^* dx dy &= 0, \\ -\nu \iint_{\Omega} (\nabla v \cdot \nabla v^*) dx dy + \frac{1}{\rho} \iint_{\Omega} \frac{\partial v^*}{\partial y} p dx dy + \iint_{\Omega} g v^* dx dy &= 0. \end{aligned}$$

ここで,  $E, F \in H^1(\Omega)$ ,  $G, H \in L^2(\Omega)$  に対して, 以下の記号を定義する。

$$\left\{ \begin{aligned} \langle E, F \rangle &= \iint_{\Omega} \left( \frac{\partial E}{\partial x} \frac{\partial F}{\partial x} + \frac{\partial E}{\partial y} \frac{\partial F}{\partial y} \right) dx dy, \\ {}_x[F, G] &= \iint_{\Omega} \frac{\partial F}{\partial x} G dx dy, \\ {}_y[F, G] &= \iint_{\Omega} \frac{\partial F}{\partial y} G dx dy, \\ (G, H) &= \iint_{\Omega} GH dx dy. \end{aligned} \right.$$

これらの記号を用いると定常 Stokes 方程式の弱形式は次のように表せる.

$$\begin{cases} -\nu \langle u^*, u \rangle + \frac{1}{\rho} x [u^*, p] + (u^*, f) = 0, \\ -\nu \langle v^*, v \rangle + \frac{1}{\rho} y [v^*, p] + (v^*, g) = 0, \\ -x [u^*, p] - y [v^*, p] = 0. \end{cases}$$

### 3.2 離散化したときの弱形式

本節では各 3 角形要素 (有限要素)  $e$  について離散化した場合の弱形式を作成する.

上で表した定常 Stokes 方程式の弱形式において,  $u, v, p, f, g, u^*, v^*, p^*$  の近似関数をそれぞれ,  $\hat{u}, \hat{v}, \hat{p}, \hat{f}, \hat{g}, \hat{u}^*, \hat{v}^*, \hat{p}^*$  として離散化すると, 以下の弱形式を得る.

$$\begin{cases} \sum_e \left\{ -\nu \langle \hat{u}^*, \hat{u} \rangle_e + \frac{1}{\rho} x [\hat{u}^*, \hat{p}]_e + (\hat{u}^*, \hat{f})_e \right\} = 0, \\ \sum_e \left\{ -\nu \langle \hat{v}^*, \hat{v} \rangle_e + \frac{1}{\rho} y [\hat{v}^*, \hat{p}]_e + (\hat{v}^*, \hat{g})_e \right\} = 0, \\ \sum_e \{ -x [\hat{u}^*, \hat{p}]_e - y [\hat{v}^*, \hat{p}]_e \} = 0. \end{cases}$$

ただし,

$$\begin{cases} \langle E, F \rangle_e = \iint_e \left( \frac{\partial E}{\partial x} \frac{\partial F}{\partial x} + \frac{\partial E}{\partial y} \frac{\partial F}{\partial y} \right) dx dy, \\ x [F, G]_e = \iint_e \frac{\partial F}{\partial x} G dx dy, \\ y [F, G]_e = \iint_e \frac{\partial F}{\partial y} G dx dy, \\ (G, H)_e = \iint_e GH dx dy. \end{cases}$$

とする. ( $E, F \in H^1(e)$ ,  $G, H \in L^2(e)$ )

この式をさらに変形して, 具体的に形が見えるようにする. その前に変形するために必要な事柄を説明する.

3 角形要素  $e$  について, 頂点を  $P_0, P_1, P_2$ , 辺  $P_1P_2$  の中点を  $P_3$ , 辺  $P_2P_0$  の中点を  $P_4$ , 辺  $P_0P_1$  の中点を  $P_5$  とする.

また, 流速  $u, v$  は各要素の頂点と中点で, 圧力  $p$  は頂点のみで考えることにする.

$P_0, P_1, P_2$  の座標を各々  $(x_0, y_0), (x_1, y_1), (x_2, y_2)$  として,  $\lambda_0, \lambda_1, \lambda_2$  を以下の式で定義する.

$$\begin{cases} \lambda_0(x, y) = \frac{(x_1y_2 - x_2y_1)}{D} + \frac{(y_1 - y_2)}{D}x + \frac{(x_2 - x_1)}{D}y, \\ \lambda_1(x, y) = \frac{(x_2y_0 - x_0y_2)}{D} + \frac{(y_2 - y_0)}{D}x + \frac{(x_0 - x_2)}{D}y, \\ \lambda_2(x, y) = \frac{(x_0y_1 - x_1y_0)}{D} + \frac{(y_0 - y_1)}{D}x + \frac{(x_1 - x_0)}{D}y. \end{cases}$$



ただし,  $D$  は  $P_0P_1$ ,  $P_0P_2$  を 2 辺とする平行四辺形の面積で, 次の式で表される.

$$D = \begin{vmatrix} 1 & x_0 & y_0 \\ 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \end{vmatrix}.$$

このとき  $\{\lambda_0, \lambda_1, \lambda_2\}$  は

$$\lambda_i(P_j) = \delta_{ij} \quad (i, j = 0, 1, 2)$$

をみたすので, 1 次の基底となっている.

また, 2 次の基底  $\{\phi_0, \phi_1, \phi_2, \phi_3, \phi_4, \phi_5\}$  は 1 次の基底を用いて, 以下のように定義できる.

$$\begin{cases} \phi_0 = \lambda_0(2\lambda_0 - 1), \\ \phi_1 = \lambda_1(2\lambda_1 - 1), \\ \phi_2 = \lambda_2(2\lambda_2 - 1), \\ \phi_3 = 4\lambda_1\lambda_2, \\ \phi_4 = 4\lambda_2\lambda_0, \\ \phi_5 = 4\lambda_0\lambda_1. \end{cases}$$

$\phi_i$  は,  $\phi_i(P_j) = \delta_{ij}$  ( $i, j = 0, 1, \dots, 5$ ) を満たしている.

$S$  を要素  $e$  の面積とすると,  $S = D/2$  であり, 面積座標の積分公式

$$\iint_e \lambda_0^l \lambda_1^m \lambda_2^n dx dy = 2S \frac{l!m!n!}{(l+m+n+2)!} \quad (l, m, n \geq 0)$$

が成立する.

定数  $\alpha_i, \beta_i$  を,

$$\begin{cases} \alpha_0 := \frac{y_1 - y_2}{D}, \\ \alpha_1 := \frac{y_2 - y_0}{D}, \\ \alpha_2 := \frac{y_0 - y_1}{D}, \\ \beta_0 := \frac{x_2 - x_1}{D}, \\ \beta_1 := \frac{x_0 - x_2}{D}, \\ \beta_2 := \frac{x_1 - x_0}{D}, \end{cases}$$

と定義する. すると,

$$\begin{aligned} \frac{\partial \phi_i}{\partial x} &= \frac{\partial}{\partial x} (\lambda_i(2\lambda_i - 1)) \\ &= (4\lambda_i - 1)\alpha_i, & (i = 0, 1, 2) \\ \frac{\partial \phi_3}{\partial x} &= \frac{\partial}{\partial x} (4\lambda_1\lambda_2) \\ &= 4\alpha_1\lambda_2 + 4\alpha_2\lambda_1, \\ \frac{\partial \phi_4}{\partial x} &= 4\alpha_2\lambda_0 + 4\alpha_0\lambda_2, \\ \frac{\partial \phi_5}{\partial x} &= 4\alpha_0\lambda_1 + 4\alpha_1\lambda_0, \end{aligned}$$

が成立する.

$\phi_i$  の  $y$  についての微分も  $\alpha_i$  を  $\beta_i$  に置き換えれば, 同様の結果を得る.  
また  $\alpha_i, \beta_i$  の定義式より

$$\begin{aligned}\alpha_0 + \alpha_1 + \alpha_2 &= \frac{1}{D} \{(y_1 - y_2) + (y_2 - y_0) + (y_0 - y_1)\} \\ &= 0, \\ \beta_0 + \beta_1 + \beta_2 &= \frac{1}{D} \{(x_2 - x_1) + (x_0 - x_2) + (x_1 - x_0)\} \\ &= 0\end{aligned}$$

となる.

$c_i$  ( $i = 0, 1, 2$ ) を以下のように定義する.

$$\begin{cases} c_0 := \alpha_1\alpha_2 + \beta_1\beta_2, \\ c_1 := \alpha_2\alpha_0 + \beta_2\beta_0, \\ c_2 := \alpha_0\alpha_1 + \beta_0\beta_1. \end{cases}$$

さらに,  $\bar{c} := c_0 + c_1 + c_2$  とする.

上に挙げた, 1 次の基底  $\lambda_i$ , 2 次の基底  $\phi_i$  を用いると  $\hat{u}, \hat{v}, \hat{p}, \hat{f}, \hat{g}, \hat{u}^*, \hat{v}^*, \hat{p}^*$  は次のように表される.

$$\begin{aligned}\hat{u} &= \sum_{i=0}^5 u_i \phi_i(x, y), & \hat{u}^* &= \sum_{i=0}^5 u_i^* \phi_i(x, y), \\ \hat{v} &= \sum_{i=0}^2 v_i \phi_i(x, y), & \hat{v}^* &= \sum_{i=0}^2 v_i^* \phi_i(x, y), \\ \hat{p} &= \sum_{i=0}^5 p_i \lambda_i(x, y), & \hat{p}^* &= \sum_{i=0}^5 p_i^* \lambda_i(x, y), \\ \hat{f} &= \sum_{i=0}^5 f_i \phi_i(x, y), & \hat{g}^* &= \sum_{i=0}^5 g_i^* \phi_i(x, y).\end{aligned}$$

ただし,  $u_i$  などは  $u$  の各頂点  $P_i$  における値を表すものとして用いている.  
この関係式を上を弱形式に代入する.

$$\begin{aligned}\langle \hat{u}^*, \hat{u} \rangle_e &= \left\langle \sum_{i=0}^5 u_i^* \phi_i, \sum_{j=0}^5 u_j \phi_j \right\rangle_e = \sum_{i=0}^5 \sum_{j=0}^5 u_i^* \langle \phi_i, \phi_j \rangle_e u_j \\ &= \hat{\mathbf{u}}_e^{*T} \hat{A}_e \hat{\mathbf{u}}_e.\end{aligned}$$

$$\begin{aligned}\langle \hat{v}^*, \hat{v} \rangle_e &= \left\langle \sum_{i=0}^2 v_i^* \phi_i, \sum_{j=0}^2 v_j \phi_j \right\rangle_e = \sum_{i=0}^2 \sum_{j=0}^2 v_i^* \langle \phi_i, \phi_j \rangle_e v_j \\ &= \hat{\mathbf{v}}_e^{*T} \hat{A}_e \hat{\mathbf{v}}_e.\end{aligned}$$

$$\begin{aligned}
x[\hat{u}^*, \hat{p}]_e &= \left[ \sum_{i=0}^5 u_i^* \phi_i, \sum_{j=0}^2 p_j \lambda_j \right]_e = \sum_{i=0}^5 \sum_{j=0}^2 u_i^* x[\phi_i, \lambda_j]_e p_j \\
&= \hat{\mathbf{u}}_e^{*T} \hat{B}_e \hat{\mathbf{p}}_e.
\end{aligned}$$

$$\begin{aligned}
y[\hat{v}^*, \hat{p}]_e &= \left[ \sum_{i=0}^5 v_i^* \phi_i, \sum_{j=0}^2 p_j \lambda_j \right]_e = \sum_{i=0}^5 \sum_{j=0}^2 v_i^* y[\phi_j, \lambda_i]_e p_j \\
&= \hat{\mathbf{v}}_e^{*T} \hat{C}_e \hat{\mathbf{p}}_e.
\end{aligned}$$

$$\begin{aligned}
x[\hat{u}, \hat{p}^*]_e &= \left[ \sum_{j=0}^5 u_j \phi_j, \sum_{i=0}^2 p_i^* \lambda_i \right]_e = \sum_{j=0}^5 \sum_{i=0}^2 p_i^* x[\phi_j, \lambda_i]_e u_j \\
&= \hat{\mathbf{p}}_e^{*T} \hat{B}_e^T \hat{\mathbf{u}}_e.
\end{aligned}$$

$$\begin{aligned}
y[\hat{v}, \hat{p}^*]_e &= \left[ \sum_{j=0}^5 v_j \phi_j, \sum_{i=0}^2 p_i^* \lambda_i \right]_e = \sum_{j=0}^5 \sum_{i=0}^2 p_i^* y[\phi_j, \lambda_i]_e v_j \\
&= \hat{\mathbf{p}}_e^{*T} \hat{C}_e^T \hat{\mathbf{v}}_e.
\end{aligned}$$

$$\begin{aligned}
(\hat{u}^*, f)_e &= \left( \sum_{i=0}^5 u_i^* \phi_i, \sum_{j=0}^5 f_j \phi_j \right)_e = \sum_{i=0}^5 \sum_{j=0}^5 u_i^* (\phi_i, \phi_j)_e f_j \\
&= \hat{\mathbf{u}}_e^{*T} \hat{D}_e \hat{\mathbf{f}}_e.
\end{aligned}$$

$$\begin{aligned}
(\hat{u}^*, g)_e &= \left( \sum_{i=0}^5 u_i^* \phi_i, \sum_{j=0}^5 g_j \phi_j \right)_e = \sum_{i=0}^5 \sum_{j=0}^5 u_i^* (\phi_i, \phi_j)_e g_j \\
&= \hat{\mathbf{u}}_e^{*T} \hat{D}_e \hat{\mathbf{g}}_e.
\end{aligned}$$

各々代入すると, 以下の方程式を得る.

$$\begin{cases} \sum_e \hat{\mathbf{u}}_e^{*T} \{ \nu \rho \hat{A}_e \hat{\mathbf{u}}_e - \hat{B}_e \hat{\mathbf{p}}_e - \rho \hat{D}_e \hat{\mathbf{f}}_e \} = 0, \\ \sum_e \hat{\mathbf{v}}_e^{*T} \{ \nu \rho \hat{A}_e \hat{\mathbf{v}}_e - \hat{C}_e \hat{\mathbf{p}}_e - \rho \hat{D}_e \hat{\mathbf{g}}_e \} = 0, \\ \sum_e \hat{\mathbf{p}}_e^{*T} \{ -\hat{B}_e^T \hat{\mathbf{u}}_e - \hat{C}_e^T \hat{\mathbf{v}}_e \} = 0. \end{cases}$$

ただし, 各  $e$  に対して,  $\hat{\mathbf{u}}_e, \hat{\mathbf{u}}_e^*, \hat{\mathbf{v}}_e, \hat{\mathbf{v}}_e^*, \hat{\mathbf{p}}_e, \hat{\mathbf{p}}_e^*, \hat{\mathbf{f}}_e, \hat{\mathbf{g}}_e$  は各頂点  $P_i$  での値を並べた列ベクトルで,  $\hat{A}_e, \hat{D}_e$  は 6 行 6 列の行列で,  $\hat{B}_e, \hat{C}_e$  は 6 行 3 列の行列であり, 各成分は以下の式で定義する.

$$\begin{cases} (\hat{A}_e)_{ij} = \langle \phi_i, \phi_j \rangle_e = \iint_e \left( \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} + \frac{\partial \phi_i}{\partial y} \frac{\partial \phi_j}{\partial y} \right) dx dy \\ (\hat{B}_e)_{ij} = x[\phi_i, \lambda_j]^e = \iint_e \frac{\partial \phi_i}{\partial x} \lambda_j dx dy \\ (\hat{C}_e)_{ij} = y[\phi_i, \lambda_j]^e = \iint_e \frac{\partial \phi_i}{\partial y} \lambda_j dx dy \\ (\hat{D}_e)_{ij} = (\phi_i, \phi_j)_e = \iint_e \phi_i \phi_j dx dy \end{cases}$$

次の節で実際に要素係数行列  $\hat{A}_e, \hat{B}_e, \hat{C}_e, \hat{D}_e$  を計算する.

### 3.3 要素係数行列の計算

この節では要素係数行列  $\hat{A}_e, \hat{B}_e, \hat{C}_e, \hat{D}_e$  を実際の計算により求めることにする.

$\hat{A}_e$  の各成分  $\langle \phi_i, \phi_j \rangle_e$  を求める.  $\hat{A}_e$  の対称性より, 上半分 ( $i \leq j$ ) のところで考えればよい.  $i = 0, 1, 2$  のとき,

$$\begin{aligned} \langle \phi_i, \phi_i \rangle_e &= \iint_e \left( \frac{\partial \phi_i}{\partial x} \right)^2 + \left( \frac{\partial \phi_i}{\partial y} \right)^2 dx dy \\ &= \iint_e ((4\lambda_i - 1)\alpha_i)^2 + ((4\lambda_i - 1)\beta_i)^2 dx dy \\ &= (\alpha_i^2 + \beta_i^2) \iint_e (16\lambda_i^2 - 8\lambda_i + 1) dx dy \\ &= (\alpha_i^2 + \beta_i^2) 2S \left[ 16 \frac{2!}{4!} - 8 \frac{1}{3!} + \frac{1}{2!} \right] \\ &= (\alpha_i^2 + \beta_i^2) S \end{aligned}$$

となる.

$i, j = 0, 1, 2 \quad i \neq j$  のとき,

$$\begin{aligned} \langle \phi_i, \phi_j \rangle_e &= \iint_e \left( \frac{\partial \phi_i}{\partial x} \right) \left( \frac{\partial \phi_j}{\partial x} \right) + \left( \frac{\partial \phi_i}{\partial y} \right) \left( \frac{\partial \phi_j}{\partial y} \right) dx dy \\ &= \iint_e ((4\lambda_i - 1)\alpha_i)((4\lambda_j - 1)\alpha_j) + ((4\lambda_i - 1)\beta_i)((4\lambda_j - 1)\beta_j) dx dy \\ &= (\alpha_i \alpha_j + \beta_i \beta_j) \iint_e (16\lambda_i \lambda_j - 4\lambda_i - 4\lambda_j + 1) dx dy \\ &= (\alpha_i \alpha_j + \beta_i \beta_j) 2S \left[ 16 \frac{1}{4!} - 4 \frac{1}{3!} - 4 \frac{1}{3!} + \frac{1}{2!} \right] \\ &= -\frac{1}{3} (\alpha_i \alpha_j + \beta_i \beta_j) S \end{aligned}$$

となる.

$$\begin{aligned}
\langle \phi_0, \phi_3 \rangle_e &= \iint_e (4\lambda_0 - 1)\alpha_0(4\alpha_1\lambda_2 + 4\alpha_2\lambda_1) + (4\lambda_0 - 1)\beta_0(4\beta_1\lambda_2 + 4\beta_2\lambda_1) dx dy \\
&= 4(\alpha_0\alpha_1 + \beta_0\beta_1) \iint_e (4\lambda_0 - 1)\lambda_2 dx dy \\
&\quad + 4(\alpha_0\alpha_2 + \beta_0\beta_2) \iint_e (4\lambda_0 - 1)\lambda_1 dx dy
\end{aligned}$$

ここで,  $i \neq j$  のとき ( $i, j = 0, 1, 2$ )

$$\begin{aligned}
\iint_e (4\lambda_i - 1)\lambda_j dx dy &= \iint_e 4\lambda_i\lambda_j - \lambda_j dx dy \\
&= 2S \left[ 4\frac{1}{4!} - \frac{1}{3!} \right] \\
&= 0
\end{aligned}$$

となるので,

$$\langle \phi_0, \phi_3 \rangle_e = 0$$

が成立する. また同様に,

$$\langle \phi_1, \phi_4 \rangle_e = \langle \phi_2, \phi_5 \rangle_e = 0$$

も成立する.

$$\begin{aligned}
\langle \phi_0, \phi_4 \rangle_e &= \iint_e (4\lambda_0 - 1)\alpha_0(4\alpha_2\lambda_0 + 4\alpha_0\lambda_2) + (4\lambda_0 - 1)\beta_0(4\beta_2\lambda_0 + 4\beta_0\lambda_2) dx dy \\
&= 4(\alpha_0\alpha_2 + \beta_0\beta_2) \iint_e (4\lambda_0 - 1)\lambda_0 dx dy \\
&\quad + 4(\alpha_0^2 + \beta_0^2) \iint_e (4\lambda_0 - 1)\lambda_2 dx dy \\
&= 4(\alpha_0\alpha_2 + \beta_0\beta_2) \iint_e (4\lambda_0 - 1)\lambda_0 dx dy \\
&= 4(\alpha_0\alpha_2 + \beta_0\beta_2) 2S \left[ 4\frac{2!}{4!} - \frac{1}{3!} \right] \\
&= \frac{4}{3}(\alpha_0\alpha_2 + \beta_0\beta_2)S \\
&= \frac{4}{3}c_1S.
\end{aligned}$$

また,  $c_i$  を用いて以下の結果も同様に得る.

$$\left\{ \begin{array}{l} \langle \phi_0, \phi_5 \rangle_e = \frac{4}{3} c_2 S, \\ \langle \phi_1, \phi_3 \rangle_e = \frac{4}{3} c_0 S, \\ \langle \phi_1, \phi_5 \rangle_e = \frac{4}{3} c_2 S, \\ \langle \phi_2, \phi_3 \rangle_e = \frac{4}{3} c_0 S, \\ \langle \phi_2, \phi_4 \rangle_e = \frac{4}{3} c_1 S. \end{array} \right.$$

$$\begin{aligned} \langle \phi_3, \phi_3 \rangle_e &= \iint_e (4\alpha_1\lambda_2 + 4\alpha_2\lambda_1)^2 + (4\beta_1\lambda_2 + 4\beta_2\lambda_1)^2 dx dy \\ &= 16(\alpha_1^2 + \beta_1^2) \iint_e \lambda_2^2 dx dy + 32(\alpha_1\alpha_2 + \beta_1\beta_2) \iint_e \lambda_1\lambda_2 dx dy \\ &\quad + 16(\alpha_2^2 + \beta_2^2) \iint_e \lambda_1^2 dx dy \\ &= \frac{8}{3} \{(\alpha_1^2 + \beta_1^2) + (\alpha_1\alpha_2 + \beta_1\beta_2) + (\alpha_2^2 + \beta_2^2)\} S \\ &= \frac{8}{3} \{\alpha_1(\alpha_1 + \alpha_2) + \beta_1(\beta_1 + \beta_2) - \alpha_2(\alpha_0 + \alpha_1) - \beta_2(\beta_0 + \beta_1)\} S \\ &= -\frac{8}{3} \{(\alpha_0\alpha_1 + \alpha_1\alpha_2 + \alpha_2\alpha_0) + (\beta_0\beta_1 + \beta_1\beta_2 + \beta_2\beta_0)\} S \\ &= -\frac{8}{3} (c_0 + c_1 + c_2) S \\ &= -\frac{8}{3} \bar{c} S. \end{aligned}$$

同様に,

$$\langle \phi_4, \phi_4 \rangle_e = \langle \phi_5, \phi_5 \rangle_e = -\frac{8}{3} \bar{c} S$$

が成立する.

$$\begin{aligned}
\langle \phi_3, \phi_4 \rangle_e &= \iint_e (4\alpha_1\lambda_2 + 4\alpha_2\lambda_1)(4\alpha_0\lambda_2 + 4\alpha_2\lambda_0) \\
&\quad + (4\beta_1\lambda_2 + 4\beta_2\lambda_1)(4\beta_0\lambda_2 + 4\beta_2\lambda_0) dx dy \\
&= 16(\alpha_0\alpha_1 + \beta_0\beta_1) \iint_e \lambda_2^2 dx dy + 16(\alpha_1\alpha_2 + \beta_1\beta_2) \iint_e \lambda_0\lambda_2 dx dy \\
&\quad + 16(\alpha_0\alpha_2 + \beta_0\beta_2) \iint_e \lambda_1\lambda_2 dx dy + 16(\alpha_2^2 + \beta_2^2) \iint_e \lambda_0\lambda_1 dx dy \\
&= \frac{4}{3} \{2(\alpha_0\alpha_1 + \beta_0\beta_1) + (\alpha_1\alpha_2 + \beta_1\beta_2) + (\alpha_0\alpha_2 + \beta_0\beta_2) + (\alpha_2^2 + \beta_2^2)\} S \\
&= \frac{4}{3} \{2(\alpha_0\alpha_1 + \beta_0\beta_1) + (\alpha_1\alpha_2 + \beta_1\beta_2) + (\alpha_0\alpha_2 + \beta_0\beta_2) \\
&\quad - \alpha_2(\alpha_0 + \alpha_1) - \beta_2(\beta_0 + \beta_1)\} S \\
&= \frac{8}{3}(\alpha_0\alpha_1 + \beta_0\beta_1)S \\
&= \frac{8}{3}c_2S.
\end{aligned}$$

$$\begin{aligned}
\langle \phi_3, \phi_5 \rangle_e &= \iint_e (4\alpha_1\lambda_2 + 4\alpha_2\lambda_1)(4\alpha_0\lambda_1 + 4\alpha_1\lambda_0) \\
&\quad + (4\beta_1\lambda_2 + 4\beta_2\lambda_1)(4\beta_0\lambda_1 + 4\beta_1\lambda_0) dx dy \\
&= 16(\alpha_0\alpha_1 + \beta_0\beta_1) \iint_e \lambda_1\lambda_2 dx dy + 16(\alpha_1^2 + \beta_1^2) \iint_e \lambda_0\lambda_2 dx dy \\
&\quad + 16(\alpha_0\alpha_2 + \beta_0\beta_2) \iint_e \lambda_1^2 dx dy + 16(\alpha_1\alpha_2 + \beta_1\beta_2) \iint_e \lambda_0\lambda_1 dx dy \\
&= \frac{4}{3} \{2(\alpha_0\alpha_2 + \beta_0\beta_2) + (\alpha_0\alpha_1 + \beta_0\beta_1) + (\alpha_1\alpha_2 + \beta_1\beta_2) + (\alpha_1^2 + \beta_1^2)\} S \\
&= \frac{8}{3}(\alpha_0\alpha_2 + \beta_0\beta_2)S \\
&= \frac{8}{3}c_1S.
\end{aligned}$$

$$\begin{aligned}
\langle \phi_4, \phi_5 \rangle_e &= \iint_e (4\alpha_0\lambda_2 + 4\alpha_2\lambda_0)(4\alpha_0\lambda_1 + 4\alpha_1\lambda_0) \\
&\quad + (4\beta_0\lambda_2 + 4\beta_2\lambda_0)(4\beta_0\lambda_1 + 4\beta_1\lambda_0) dx dy \\
&= 16(\alpha_0^2 + \beta_0^2) \iint_e \lambda_1\lambda_2 dx dy + 16(\alpha_0\alpha_1 + \beta_0\beta_1) \iint_e \lambda_0\lambda_2 dx dy \\
&\quad + 16(\alpha_0\alpha_2 + \beta_0\beta_2) \iint_e \lambda_0\lambda_1 dx dy + 16(\alpha_1\alpha_2 + \beta_1\beta_2) \iint_e \lambda_0^2 dx dy \\
&= \frac{4}{3} \{2(\alpha_1\alpha_2 + \beta_1\beta_2) + (\alpha_0\alpha_2 + \beta_0\beta_2) + (\alpha_1\alpha_0 + \beta_1\beta_0) + (\alpha_0^2 + \beta_0^2)\} S \\
&= \frac{8}{3}(\alpha_1\alpha_2 + \beta_1\beta_2)S \\
&= \frac{8}{3}c_0S.
\end{aligned}$$

これで  $\hat{A}_e$  の各成分  $\langle \phi_i, \phi_j \rangle_e$  を求めることができた.

次に  $\hat{B}_e$  の各成分  $x[\phi_i, \lambda_j]^e$  を求める.

$i = 0, 1, 2$  のとき,

$$\begin{aligned} x[\phi_i, \lambda_i]^e &= \iint_e (4\lambda_i - 1)\alpha_i \lambda_i dx dy \\ &= \alpha_i \iint_e (4\lambda_i^2 - \lambda_i) dx dy \\ &= \alpha_i 2S \left[ 4 \frac{2!}{4!} - \frac{1}{3!} \right] \\ &= \frac{1}{3} \alpha_i S \end{aligned}$$

が成立する.

$i, j = 0, 1, 2 \quad i \neq j$  のとき,

$$\begin{aligned} x[\phi_i, \lambda_j]^e &= \iint_e (4\lambda_i - 1)\alpha_i \lambda_j dx dy \\ &= \alpha_i \iint_e (4\lambda_i \lambda_j - \lambda_j) dx dy \\ &= \alpha_i 2S \left[ 4 \frac{1}{4!} - \frac{1}{3!} \right] \\ &= 0 \end{aligned}$$

が成立する.

$$\begin{aligned} x[\phi_3, \lambda_0]^e &= \iint_e (4\alpha_1 \lambda_2 - 4\alpha_2 \lambda_1) \lambda_0 dx dy \\ &= 4\alpha_1 \iint_e \lambda_0 \lambda_2 dx dy + 4\alpha_2 \iint_e \lambda_1 \lambda_0 dx dy \\ &= \frac{1}{3} (\alpha_1 + \alpha_2) S. \end{aligned}$$

同様に,

$$\begin{cases} x[\phi_4, \lambda_1]^e = \frac{1}{3} (\alpha_2 + \alpha_0) S. \\ x[\phi_5, \lambda_2]^e = \frac{1}{3} (\alpha_0 + \alpha_1) S. \end{cases}$$

が成立する.

$$\begin{aligned} x[\phi_3, \lambda_1]^e &= \iint_e (4\alpha_1 \lambda_2 - 4\alpha_2 \lambda_1) \lambda_1 dx dy \\ &= 4\alpha_1 \iint_e \lambda_1 \lambda_2 dx dy + 4\alpha_2 \iint_e \lambda_1^2 dx dy \\ &= \frac{1}{3} (\alpha_1 + 2\alpha_2) S. \end{aligned}$$



同様に,

$$\begin{cases} x[\phi_3, \lambda_2]^e = \frac{1}{3}(2\alpha_1 + \alpha_2)S, \\ x[\phi_4, \lambda_0]^e = \frac{1}{3}(2\alpha_2 + \alpha_0)S, \\ x[\phi_4, \lambda_2]^e = \frac{1}{3}(\alpha_2 + 2\alpha_0)S, \\ x[\phi_5, \lambda_0]^e = \frac{1}{3}(\alpha_0 + 2\alpha_1)S, \\ x[\phi_5, \lambda_1]^e = \frac{1}{3}(2\alpha_0 + \alpha_1)S \end{cases}$$

が成立する.

これで  $\hat{B}_e$  の各成分  $x[\phi_i, \lambda_j]^e$  を求めることができた.

$\hat{C}_e$  の各成分  $y[\phi_i, \lambda_j]^e$  については,  $\hat{B}_e$  の各成分で  $\alpha_i$  を  $\beta_i$  ( $i = 0, 1, 2$ ) と置き換えれば求まる.

最後に  $\hat{D}_e$  の各成分  $(\phi_i, \phi_j)_e$  を求める.

$\hat{D}_e$  の対称性より, 上半分 ( $i \leq j$ ) のところで考えればよい.  
 $i = 0, 1, 2$  のとき,

$$\begin{aligned} (\phi_i, \phi_i)_e &= \iint_e (2\lambda_i^2 - \lambda_i)^2 dx dy \\ &= \iint_e (4\lambda_i^4 - 4\lambda_i^3 + \lambda_i^2) dx dy \\ &= 2S \left[ 4\frac{4!}{6!} - 4\frac{3!}{5!} + \frac{2!}{4!} \right] \\ &= \frac{1}{30}S \end{aligned}$$

が成立する.

$i, j = 0, 1, 2$   $i \neq j$  のとき,

$$\begin{aligned} (\phi_i, \phi_j)_e &= \iint_e (2\lambda_i^2 - \lambda_i)(2\lambda_j^2 - \lambda_j) dx dy \\ &= \iint_e (4\lambda_i^2\lambda_j^2 - 2\lambda_i^2\lambda_j - 2\lambda_i\lambda_j^2 + \lambda_i\lambda_j) dx dy \\ &= 2S \left[ 4\frac{2!2!}{6!} - 2\frac{2!}{5!} - 2\frac{2!}{5!} + \frac{1}{4!} \right] \\ &= -\frac{1}{180}S \end{aligned}$$

が成立する.

$$\begin{aligned}
(\phi_0, \phi_3)_e &= \iint_e (2\lambda_0^2 - \lambda_0)(4\lambda_1\lambda_2) dx dy \\
&= \iint_e (8\lambda_0^2\lambda_1\lambda_2 - 4\lambda_0\lambda_1\lambda_2) dx dy \\
&= 2S \left[ 8\frac{2!}{6!} - 4\frac{1!}{5!} \right] \\
&= -\frac{1}{45}S.
\end{aligned}$$

同様に,

$$(\phi_1, \phi_4)_e = (\phi_2, \phi_5)_e = -\frac{1}{45}S$$

が成立する.

$$\begin{aligned}
(\phi_0, \phi_4)_e &= \iint_e (2\lambda_0^2 - \lambda_0)(4\lambda_2\lambda_0) dx dy \\
&= \iint_e (8\lambda_0^3\lambda_2 - 4\lambda_0^2\lambda_2) dx dy \\
&= 2S \left[ 8\frac{3!}{6!} - 4\frac{2!}{5!} \right] \\
&= 0.
\end{aligned}$$

同様に,

$$(\phi_0, \phi_5)_e = (\phi_1, \phi_3)_e = (\phi_1, \phi_5)_e = (\phi_2, \phi_3)_e = (\phi_2, \phi_4)_e = 0$$

が成立する.

$$\begin{aligned}
(\phi_3, \phi_3)_e &= \iint_e (4\lambda_1\lambda_2)^2 dx dy \\
&= 16 \iint_e \lambda_1^2\lambda_2^2 dx dy \\
&= 16 \left( 2S\frac{2!2!}{6!} \right) \\
&= \frac{8}{45}S.
\end{aligned}$$

同様に,

$$(\phi_4, \phi_4)_e = (\phi_5, \phi_5)_e = \frac{8}{45}S$$

が成立する.

$$\begin{aligned}
(\phi_3, \phi_4)_e &= \iint_e (4\lambda_1\lambda_2)(4\lambda_2\lambda_0) dx dy \\
&= 16 \iint_e \lambda_0\lambda_1\lambda_2^2 dx dy \\
&= 16 \left( 2S \frac{2!}{6!} \right) \\
&= \frac{4}{45}S.
\end{aligned}$$

同様に,

$$(\phi_3, \phi_5)_e = (\phi_4, \phi_5)_e = \frac{4}{45}S$$

が成立する.

これで  $\hat{D}_e$  の各成分  $(\phi_i, \phi_j)_e$  を求めることができた.

よって, 行列  $\hat{A}_e, \hat{B}_e, \hat{C}_e, \hat{D}_e$  の各成分を求めることができた.

計算結果をまとめる

要素係数行列  $\hat{A}_e, \hat{B}_e, \hat{C}_e, \hat{D}_e$  を表示する.

$$D = \begin{vmatrix} 1 & x_0 & y_0 \\ 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \end{vmatrix},$$

$$S = \frac{1}{2}D,$$

$$\left\{ \begin{array}{l}
\alpha_0 := \frac{y_1 - y_2}{D}, \\
\alpha_1 := \frac{y_2 - y_0}{D}, \\
\alpha_2 := \frac{y_0 - y_1}{D}, \\
\beta_0 := \frac{x_2 - x_1}{D}, \\
\beta_1 := \frac{x_0 - x_2}{D}, \\
\beta_2 := \frac{x_1 - x_0}{D}, \\
c_0 := \alpha_1\alpha_2 + \beta_1\beta_2, \\
c_1 := \alpha_2\alpha_0 + \beta_2\beta_0, \\
c_2 := \alpha_0\alpha_1 + \beta_0\beta_1, \\
\bar{c} := c_0 + c_1 + c_2,
\end{array} \right.$$

とすると,

$$\hat{A}_e = \frac{S}{3} \begin{bmatrix} 3(\alpha_0^2 + \beta_0^2) & -c_2 & -c_1 & 0 & 4c_1 & 4c_2 \\ -c_2 & 3(\alpha_1^2 + \beta_1^2) & -c_0 & 4c_0 & 0 & 4c_2 \\ -c_1 & -c_0 & 3(\alpha_2^2 + \beta_2^2) & 4c_0 & 4c_1 & 0 \\ 0 & 4c_0 & 4c_0 & -8\bar{c} & 8c_2 & 8c_1 \\ 4c_1 & 0 & 4c_1 & 8c_2 & -8\bar{c} & 8c_0 \\ 4c_2 & 4c_2 & 0 & 8c_1 & 8c_0 & -8\bar{c} \end{bmatrix}$$

$$\hat{B}_e = \frac{S}{3} \begin{bmatrix} \alpha_0 & 0 & 0 \\ 0 & \alpha_1 & 0 \\ 0 & 0 & \alpha_2 \\ \alpha_1 + \alpha_2 & \alpha_1 + 2\alpha_2 & 2\alpha_1 + \alpha_2 \\ 2\alpha_2 + \alpha_0 & \alpha_2 + \alpha_0 & \alpha_2 + 2\alpha_0 \\ \alpha_0 + 2\alpha_1 & 2\alpha_0 + \alpha_1 & \alpha_0 + \alpha_1 \end{bmatrix}$$

$$\hat{C}_e = \frac{S}{3} \begin{bmatrix} \beta_0 & 0 & 0 \\ 0 & \beta_1 & 0 \\ 0 & 0 & \beta_2 \\ \beta_1 + \beta_2 & \beta_1 + 2\beta_2 & 2\beta_1 + \beta_2 \\ 2\beta_2 + \beta_0 & \beta_2 + \beta_0 & \beta_2 + 2\beta_0 \\ \beta_0 + 2\beta_1 & 2\beta_0 + \beta_1 & \beta_0 + \beta_1 \end{bmatrix}$$

$$\hat{D}_e = \frac{S}{180} \begin{bmatrix} 6 & -1 & -1 & -4 & 0 & 0 \\ -1 & 6 & -1 & 0 & -4 & 0 \\ -1 & -1 & 6 & 0 & 0 & -4 \\ -4 & 0 & 0 & 32 & 16 & 16 \\ 0 & -4 & 0 & 16 & 32 & 16 \\ 0 & 0 & -4 & 16 & 16 & 32 \end{bmatrix}$$

### 3.4 近似方程式の作成

領域を3角形分割したとき,各要素と,各要素の頂点と各辺の midpoint (節点)に0から番号付けをしていく.この番号を要素番号,全体節点番号という.ただし第0全体節点は要素の頂点にとることにする.

流速は全ての節点(要素の頂点と辺の midpoint)で値を計算するが,圧力は要素の頂点でのみ値を計算する.

そこで,第*i*全体節点における未知数の個数を第*i*全体節点における自由度と呼び*f<sub>i</sub>*で書くことにする.(大まかに言うと,頂点では自由度3, midpointでは自由度2)第0全体節点は頂点としたので,*f<sub>0</sub>* = 3である.全節点の自由度を足し合わせた値が未知数の総数になる.未知数の総数を*N*とおく.

ここで,第*i*節点における累積節点番号 *cum*[*i*] を以下の式で定義する.

$$cum[0] = 0, \quad cum[i] = \sum_{n=0}^{i-1} f_n \quad (i \leq 1).$$

そして、全節点での  $u, v, p$  (ただし  $p$  は要素の頂点のみ) を並べた  $N$  次ベクトルを考える. 第  $i$  全体節点での  $u$  の値がこのベクトルの第  $\text{cum}[i]$  成分に,  $v$  の値が  $\text{cum}[i] + 1$  成分に,  $p$  の値が  $\text{cum}[i] + 2$  成分に対応する. このベクトルに関する連立 1 次方程式を作成する.

$\hat{A}_e$  を  $N$  次に拡大した行列を  $A_e^*$  とする. 局所節点番号 (前に挙げた, 要素における節点  $P_0$  から  $P_5$  のこと) と全体節点番号の対応付けをもとに  $\hat{A}_e$  の各成分を  $A_e^*$  に対応させる.

そして、全要素について足し合わせた行列を  $A$  とする. 他の行列やベクトルも同様に拡大して全要素について和をとると, 以下のような式を得る. (直接剛性法)

$$\begin{cases} \mathbf{u}^{*T}(\nu\rho A\mathbf{u} - B\mathbf{p} - \rho D\mathbf{f}) = 0 \\ \mathbf{v}^{*T}(\nu\rho A\mathbf{v} - C\mathbf{p} - \rho D\mathbf{g}) = 0, \\ \mathbf{p}^{*T}(-B^T\mathbf{u} - C^T\mathbf{v}) = 0. \end{cases}$$

ただし, 各記号については  $A$  に準じる.

さらに,  $\mathbf{u}^{*T}, \mathbf{v}^{*T}, \mathbf{p}^{*T}$  の任意性より, 以下の近似方程式を得る.

$$\begin{cases} \nu\rho A\mathbf{u} - B\mathbf{p} - \rho D\mathbf{f} = 0, \\ \nu\rho A\mathbf{v} - C\mathbf{p} - \rho D\mathbf{g} = 0, \\ -B^T\mathbf{u} - C^T\mathbf{v} = 0. \end{cases}$$

これは,  $\mathbf{u}, \mathbf{v}, \mathbf{p}$  の連立 1 次方程式であり, 境界条件を考慮してこれを解けば Stokes 方程式の数値解を得る.

## 4 定常 Navier-Stokes 方程式

### 4.1 非線形項の弱定式化

非線形項  $(\mathbf{u} \cdot \Delta)\mathbf{u}$  の弱形式について考える.

$$(\mathbf{u} \cdot \Delta)\mathbf{u} = \left( u \frac{\partial}{\partial x} + v \frac{\partial}{\partial y} \right) \mathbf{u} = (uu_x + vu_y, uv_x + vv_y)^T.$$

第1成分について  $u^*$  を掛けて  $\Omega$  上で積分すると

$$\iint_{\Omega} u^*(uu_x + vu_y) dx dy = \iint_{\Omega} u^*uu_x dx dy + \iint_{\Omega} u^*vu_y dx dy.$$

第2成分について  $v^*$  を掛けて  $\Omega$  上で積分すると

$$\iint_{\Omega} v^*(uv_x + vv_y) dx dy = \iint_{\Omega} v^*uv_x dx dy + \iint_{\Omega} v^*vv_y dx dy.$$

この式を  $u, v, u^*, v^*$  の近似関数  $\hat{u}, \hat{v}, \hat{u}^*, \hat{v}^*$

$$\begin{aligned} \hat{u} &= \sum_{i=0}^5 u_i \phi_i(x, y), & \hat{u}^* &= \sum_{i=0}^5 u_i^* \phi_i(x, y), \\ \hat{v} &= \sum_{i=0}^5 v_i \phi_i(x, y), & \hat{v}^* &= \sum_{i=0}^5 v_i^* \phi_i(x, y). \end{aligned}$$

を用いて離散化する.

以下の記号を定義する.

$$\begin{aligned} \kappa_{ijk}^x &:= \iint_e \phi_i \phi_j \frac{\partial \phi_k}{\partial x} dx dy, & \kappa_{ijk}^y &:= \iint_e \phi_i \phi_j \frac{\partial \phi_k}{\partial y} dx dy, \\ S_{ij}^{x1} &:= \sum_{k=0}^5 \kappa_{ikj}^x u_k, & S_{ij}^{x2} &:= \sum_{k=0}^5 \kappa_{ijk}^x u_k, \\ S_{ij}^{y1} &:= \sum_{k=0}^5 \kappa_{ikj}^y u_k, & S_{ij}^{y2} &:= \sum_{k=0}^5 \kappa_{ijk}^y u_k, \\ T_{ij}^{x1} &:= \sum_{k=0}^5 \kappa_{ikj}^x v_k, & T_{ij}^{x2} &:= \sum_{k=0}^5 \kappa_{ijk}^x v_k, \\ T_{ij}^{y1} &:= \sum_{k=0}^5 \kappa_{ikj}^y v_k, & T_{ij}^{y2} &:= \sum_{k=0}^5 \kappa_{ijk}^y v_k, \end{aligned}$$

$$\hat{S}_e^{x1} := [S_{ij}^{x1}], \quad \hat{S}_e^{x2} := [S_{ij}^{x2}], \quad \hat{S}_e^{y1} := [S_{ij}^{y1}], \quad \hat{S}_e^{y2} := [S_{ij}^{y2}],$$

$$\hat{T}_e^{x1} := [T_{ij}^{x1}], \quad \hat{T}_e^{x2} := [T_{ij}^{x2}], \quad \hat{T}_e^{y1} := [T_{ij}^{y1}], \quad \hat{T}_e^{y2} := [T_{ij}^{y2}].$$

$k$  を固定して, 行列  $\kappa_{ijk}^x, \kappa_{ijk}^y$  を実際に計算する. 計算には Mathematica を用いた. 行列を求めるプログラムは (6) 節を参照のこと.

まず,  $\kappa_{ijk}^x$  を求める.

$$\kappa_{ij0}^x = \frac{\alpha_0 S}{1260} \begin{bmatrix} 78 & -9 & -9 & 12 & 48 & 48 \\ -9 & -18 & 11 & -16 & -20 & -32 \\ -9 & 11 & -18 & -16 & -32 & -20 \\ 12 & -16 & -16 & -96 & 16 & 16 \\ 48 & -20 & -32 & 16 & 160 & 80 \\ 48 & -32 & -20 & 16 & 80 & 160 \end{bmatrix}$$

$$\kappa_{ij1}^x = \frac{\alpha_1 S}{1260} \begin{bmatrix} -18 & -9 & 11 & -20 & -16 & -32 \\ -9 & 78 & -9 & 48 & 12 & 48 \\ 11 & -9 & -18 & -32 & -16 & -20 \\ -20 & 48 & -32 & 160 & 16 & 80 \\ -16 & 12 & -16 & 16 & -96 & 16 \\ -32 & 48 & -20 & 80 & 16 & 160 \end{bmatrix}$$

$$\kappa_{ij2}^x = \frac{\alpha_2 S}{1260} \begin{bmatrix} -18 & 11 & -9 & -20 & -32 & -16 \\ 11 & -18 & -9 & -32 & -20 & -16 \\ -9 & -9 & 78 & 48 & 48 & 12 \\ -20 & -32 & 48 & 160 & 80 & 16 \\ -32 & -20 & 48 & 80 & 160 & 16 \\ -16 & -16 & 12 & 16 & 16 & -96 \end{bmatrix}$$

$$\kappa_{ij3}^x = \frac{\alpha_1 S}{1260} \begin{bmatrix} 24 & 4 & -16 & -48 & -32 & -16 \\ 4 & 24 & -16 & -32 & -48 & -16 \\ -16 & -16 & 120 & 48 & 48 & -16 \\ -48 & -32 & 48 & 384 & 192 & 128 \\ -32 & -48 & 48 & 192 & 384 & 128 \\ -16 & -16 & -16 & 128 & 128 & 128 \end{bmatrix} + \frac{\alpha_2 S}{1260} \begin{bmatrix} 24 & -16 & 4 & -48 & -16 & -32 \\ -16 & 120 & -16 & 48 & -16 & 48 \\ 4 & -16 & 24 & -32 & -16 & -48 \\ -48 & 48 & -32 & 384 & 128 & 192 \\ -16 & -16 & -16 & 128 & 128 & 128 \\ -32 & 48 & -48 & 192 & 128 & 384 \end{bmatrix}$$

$$\kappa_{ij4}^x = \frac{\alpha_0 S}{1260} \begin{bmatrix} 24 & 4 & -16 & -48 & -32 & -16 \\ 4 & 24 & -16 & -32 & -48 & -16 \\ -16 & -16 & 120 & 48 & 48 & -16 \\ -48 & -32 & 48 & 384 & 192 & 128 \\ -32 & -48 & 48 & 192 & 384 & 128 \\ -16 & -16 & -16 & 128 & 128 & 128 \end{bmatrix} + \frac{\alpha_2 S}{1260} \begin{bmatrix} 120 & -16 & -16 & -16 & 48 & 48 \\ -16 & 24 & 4 & -16 & -48 & -32 \\ -16 & 4 & 24 & -16 & -32 & -48 \\ -16 & -16 & -16 & 128 & 128 & 128 \\ 48 & -48 & -32 & 128 & 384 & 192 \\ 48 & -32 & -48 & 128 & 192 & 384 \end{bmatrix}$$

$$\kappa_{ij5}^x = \frac{\alpha_0 S}{1260} \begin{bmatrix} 24 & -16 & 4 & -48 & -16 & -32 \\ -16 & 120 & -16 & 48 & -16 & 48 \\ 4 & -16 & 24 & -32 & -16 & -48 \\ -48 & 48 & -32 & 384 & 128 & 192 \\ -16 & -16 & -16 & 128 & 128 & 128 \\ -32 & 48 & -48 & 192 & 128 & 384 \end{bmatrix} + \frac{\alpha_1 S}{1260} \begin{bmatrix} 120 & -16 & -16 & -16 & 48 & 48 \\ -16 & 24 & 4 & -16 & -48 & -32 \\ -16 & 4 & 24 & -16 & -32 & -48 \\ -16 & -16 & -16 & 128 & 128 & 128 \\ 48 & -48 & -32 & 128 & 384 & 192 \\ 48 & -32 & -48 & 128 & 192 & 384 \end{bmatrix}$$

行列  $\kappa_{ijk}^y$  については, 行列  $\kappa_{ijk}^x$  の係数  $\alpha_i$  を  $\beta_i$  ( $i = 0, 1, 2$ ) と置き換えれば求まる.  
 まず, 第 1 成分の第 1 項について離散化した式を考える.

$$\begin{aligned}
 \sum_e \iint_e \hat{u}^* \hat{u}_x dx dy &= \sum_e \iint_e \sum_{i=0}^5 u_i^* \phi_i \sum_{j=0}^5 u_j \phi_j \sum_{k=0}^5 u_k \frac{\partial \phi_k}{\partial x} dx dy \\
 &= \sum_e \left\{ \sum_{i,j,k=0}^5 u_i^* \left( \iint_e \phi_i \phi_j \frac{\partial \phi_k}{\partial x} dx dy \right) u_j u_k \right\} \\
 &= \sum_e \left\{ \sum_{i,j,k=0}^5 u_i^* \kappa_{ijk}^x u_j u_k \right\} \\
 &= \sum_e \left\{ \sum_{i=0}^5 u_i^* \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^x u_k \right) u_j \right\} \\
 &= \sum_e \left\{ \sum_{i=0}^5 u_i^* \sum_{j=0}^5 S_{ij}^{x2} u_j \right\} \\
 &= \sum_e \hat{\mathbf{u}}_e^{*T} \hat{S}_e^{x2} \hat{\mathbf{u}}_e.
 \end{aligned}$$

第 1 成分の第 2 項についても同様に,

$$\begin{aligned}
 \sum_e \iint_e \hat{u}^* \hat{v}_y dx dy &= \sum_e \iint_e \sum_{i=0}^5 u_i^* \phi_i \sum_{j=0}^5 v_j \phi_j \sum_{k=0}^5 u_k \frac{\partial \phi_k}{\partial y} dx dy \\
 &= \sum_e \left\{ \sum_{i,j,k=0}^5 u_i^* \left( \iint_e \phi_i \phi_j \frac{\partial \phi_k}{\partial y} dx dy \right) v_j u_k \right\} \\
 &= \sum_e \left\{ \sum_{i=0}^5 u_i^* \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^y u_k \right) v_j \right\} \\
 &= \sum_e \left\{ \sum_{i=0}^5 u_i^* \sum_{j=0}^5 S_{ij}^{y2} v_j \right\} \\
 &= \sum_e \hat{\mathbf{u}}_e^{*T} \hat{S}_e^{y2} \hat{\mathbf{v}}_e.
 \end{aligned}$$



第2成分の第1項についても同様に,

$$\begin{aligned}
\sum_e \iint_e \hat{v}^* \hat{u} \hat{v}_x dx dy &= \sum_e \iint_e \sum_{i=0}^5 v_i^* \phi_i \sum_{j=0}^5 u_j \phi_j \sum_{k=0}^5 v_k \frac{\partial \phi_k}{\partial x} dx dy \\
&= \sum_e \left\{ \sum_{i,j,k=0}^5 v_i^* \left( \iint_e \phi_i \phi_j \frac{\partial \phi_k}{\partial x} dx dy \right) u_j v_k \right\} \\
&= \sum_e \left\{ \sum_{i=0}^5 v_i^* \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^x v_k \right) u_j \right\} \\
&= \sum_e \left\{ \sum_{i=0}^5 v_i^* \sum_{j=0}^5 T_{ij}^{x2} u_j \right\} \\
&= \sum_e \hat{\mathbf{v}}_e^{*T} \hat{T}_e^{x2} \hat{\mathbf{u}}_e.
\end{aligned}$$

第2成分の第2項についても同様に,

$$\begin{aligned}
\sum_e \iint_e \hat{v}^* \hat{v} \hat{v}_y dx dy &= \sum_e \iint_e \sum_{i=0}^5 v_i^* \phi_i \sum_{j=0}^5 v_j \phi_j \sum_{k=0}^5 v_k \frac{\partial \phi_k}{\partial y} dx dy \\
&= \sum_e \left\{ \sum_{i,j,k=0}^5 v_i^* \left( \iint_e \phi_i \phi_j \frac{\partial \phi_k}{\partial y} dx dy \right) v_j v_k \right\} \\
&= \sum_e \left\{ \sum_{i=0}^5 v_i^* \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^y v_k \right) v_j \right\} \\
&= \sum_e \left\{ \sum_{i=0}^5 v_i^* \sum_{j=0}^5 T_{ij}^{y2} v_j \right\} \\
&= \sum_e \hat{\mathbf{v}}_e^{*T} \hat{T}_e^{y2} \hat{\mathbf{v}}_e.
\end{aligned}$$

これより Navier-Stokes 方程式を離散化したときの弱形式は,

$$\begin{cases} \sum_e \hat{\mathbf{u}}_e^{*T} \left\{ \nu \rho \hat{A}_e \hat{\mathbf{u}}_e - \hat{B}_e \hat{\mathbf{p}}_e - \rho \hat{D}_e \hat{\mathbf{f}}_e + (\hat{S}_e^{x2} \hat{\mathbf{u}}_e + \hat{S}_e^{y2} \hat{\mathbf{v}}_e) \right\} = 0, \\ \sum_e \hat{\mathbf{v}}_e^{*T} \left\{ \nu \rho \hat{A}_e \hat{\mathbf{v}}_e - \hat{C}_e \hat{\mathbf{p}}_e - \rho \hat{D}_e \hat{\mathbf{g}}_e + (\hat{T}_e^{x2} \hat{\mathbf{u}}_e + \hat{T}_e^{y2} \hat{\mathbf{v}}_e) \right\} = 0, \\ \sum_e \hat{\mathbf{p}}_e^{*T} \left\{ -\hat{B}_e^T \hat{\mathbf{u}}_e - \hat{C}_e^T \hat{\mathbf{v}}_e \right\} = 0. \end{cases}$$

$\hat{S}_e^{x1}, \hat{S}_e^{x2}, \hat{S}_e^{y1}, \hat{S}_e^{y2}, \hat{T}_e^{x1}, \hat{T}_e^{x2}, \hat{T}_e^{y1}, \hat{T}_e^{y2}$  を拡大して全要素について和をとった行列を  $S^{x1}, S^{x2}, S^{y1}, S^{y2}, T^{x1}, T^{x2}, T^{y1}, T^{y2}$  とする.

すると, 直接剛性法により以下のような近似方程式が得られる.

$$\begin{cases} \nu \rho A \mathbf{u} - B \mathbf{p} - \rho D \mathbf{f} + \rho (S^{x2} \mathbf{u} + S^{y2} \mathbf{v}) = 0, \\ \nu \rho A \mathbf{v} - C \mathbf{p} - \rho D \mathbf{g} + \rho (T^{x2} \mathbf{u} + T^{y2} \mathbf{v}) = 0, \\ -B^T \mathbf{u} - C^T \mathbf{v} = 0. \end{cases}$$

## 4.2 Newton 法

近似方程式の非線形項  $\hat{S}_e^{x2}\hat{\mathbf{u}}_e$ ,  $\hat{S}_e^{y2}\hat{\mathbf{v}}_e$ ,  $\hat{T}_e^{x2}\hat{\mathbf{u}}_e$ ,  $\hat{T}_e^{y2}\hat{\mathbf{v}}_e$  において,  $u_i, v_i$  に  $u_i + \Delta u_i, v_i + \Delta v_i$  を代入する.

$$\Delta \hat{\mathbf{u}}_e = [\Delta u_i], \quad \Delta \hat{\mathbf{v}}_e = [\Delta v_i]$$

とおく.

また,  $\Delta \hat{\mathbf{u}}_e, \Delta \hat{\mathbf{v}}_e$  の 2 次の項は切り捨てることにする.

$$\begin{aligned} & \hat{S}_e^{x2}\hat{\mathbf{u}}_e \Big|_{\hat{\mathbf{u}}_e = \hat{\mathbf{u}}_e + \Delta \hat{\mathbf{u}}_e} \\ &= \sum_{i=0}^5 u_i^* \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^x (u_k - \Delta u_k) \right) (u_j - \Delta u_j) \\ &= \sum_{i=0}^5 u_i^* \left\{ \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^x u_k \right) u_j - \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^x u_k \right) \Delta u_j \right. \\ & \quad \left. - \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^x \Delta u_k \right) u_j + \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^x \Delta u_k \right) \Delta u_j \right\} \\ &= \sum_{i=0}^5 u_i^* \left\{ \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^x u_k \right) u_j - \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^x u_k \right) \Delta u_j - \sum_{k=0}^5 \left( \sum_{j=0}^5 \kappa_{ijk}^x u_j \right) \Delta u_k \right\} \\ &= \sum_{i=0}^5 u_i^* \left\{ \sum_{j=0}^5 S_{ij}^{x2} u_j - \sum_{j=0}^5 S_{ij}^{x2} \Delta u_j - \sum_{k=0}^5 S_{ik}^{x1} \Delta u_k \right\} \\ &= \sum_{i=0}^5 u_i^* \sum_{j=0}^5 \{ S_{ij}^{x2} u_j - S_{ij}^{x2} \Delta u_j - S_{ij}^{x1} \Delta u_j \} \\ &= \hat{\mathbf{u}}_e^{*T} \hat{S}_e^{x2} \hat{\mathbf{u}}_e - \hat{\mathbf{u}}_e^{*T} \hat{S}_e^{x2} \Delta \hat{\mathbf{u}}_e - \hat{\mathbf{u}}_e^{*T} \hat{S}_e^{x1} \Delta \hat{\mathbf{u}}_e. \end{aligned}$$

$$\begin{aligned}
& \hat{S}_e^{y2} \hat{\mathbf{v}}_e \Big|_{\hat{\mathbf{u}}_e = \hat{\mathbf{u}}_e + \Delta \hat{\mathbf{u}}_e, \hat{\mathbf{v}}_e = \hat{\mathbf{v}}_e + \Delta \hat{\mathbf{v}}_e} \\
&= \sum_{i=0}^5 u_i^* \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^y (u_k - \Delta u_k) \right) (v_j - \Delta v_j) \\
&= \sum_{i=0}^5 u_i^* \left\{ \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^y u_k \right) v_j - \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^y u_k \right) \Delta v_j \right. \\
&\quad \left. - \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^y \Delta u_k \right) v_j + \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^y \Delta u_k \right) \Delta v_j \right\} \\
&= \sum_{i=0}^5 u_i^* \left\{ \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^y u_k \right) v_j - \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^y u_k \right) \Delta v_j - \sum_{k=0}^5 \left( \sum_{j=0}^5 \kappa_{ijk}^y v_j \right) \Delta u_k \right\} \\
&= \sum_{i=0}^5 u_i^* \left\{ \sum_{j=0}^5 S_{ij}^{y2} v_j - \sum_{j=0}^5 S_{ij}^{y2} \Delta v_j - \sum_{k=0}^5 T_{ik}^{y1} \Delta u_k \right\} \\
&= \sum_{i=0}^5 u_i^* \sum_{j=0}^5 \{ S_{ij}^{y2} v_j - S_{ij}^{y2} \Delta v_j - T_{ij}^{y1} \Delta u_j \} \\
&= \hat{\mathbf{u}}_e^{*T} \hat{S}_e^{y2} \hat{\mathbf{v}}_e - \hat{\mathbf{u}}_e^{*T} \hat{S}_e^{y2} \Delta \hat{\mathbf{v}}_e - \hat{\mathbf{u}}_e^{*T} \hat{T}_e^{y1} \Delta \hat{\mathbf{u}}_e.
\end{aligned}$$

$$\begin{aligned}
& \hat{T}_e^{x2} \hat{\mathbf{u}}_e \Big|_{\hat{\mathbf{u}}_e = \hat{\mathbf{u}}_e + \Delta \hat{\mathbf{u}}_e, \hat{\mathbf{v}}_e = \hat{\mathbf{v}}_e + \Delta \hat{\mathbf{v}}_e} \\
&= \sum_{i=0}^5 v_i^* \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^x (v_k - \Delta v_k) \right) (u_j - \Delta u_j) \\
&= \sum_{i=0}^5 v_i^* \left\{ \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^x v_k \right) u_j - \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^x v_k \right) \Delta u_j \right. \\
&\quad \left. - \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^x \Delta v_k \right) u_j + \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^x \Delta v_k \right) \Delta u_j \right\} \\
&= \sum_{i=0}^5 v_i^* \left\{ \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^x v_k \right) u_j - \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^x v_k \right) \Delta u_j - \sum_{k=0}^5 \left( \sum_{j=0}^5 \kappa_{ijk}^x u_j \right) \Delta v_k \right\} \\
&= \sum_{i=0}^5 v_i^* \left\{ \sum_{j=0}^5 T_{ij}^{x2} u_j - \sum_{j=0}^5 T_{ij}^{x2} \Delta u_j - \sum_{k=0}^5 S_{ik}^{x1} \Delta v_k \right\} \\
&= \sum_{i=0}^5 v_i^* \sum_{j=0}^5 \{ T_{ij}^{x2} u_j - T_{ij}^{x2} \Delta u_j - S_{ij}^{x1} \Delta v_j \} \\
&= \hat{\mathbf{v}}_e^{*T} \hat{T}_e^{x2} \hat{\mathbf{u}}_e - \hat{\mathbf{v}}_e^{*T} \hat{T}_e^{x2} \Delta \hat{\mathbf{u}}_e - \hat{\mathbf{v}}_e^{*T} \hat{S}_e^{x1} \Delta \hat{\mathbf{v}}_e.
\end{aligned}$$

$$\begin{aligned}
& \hat{T}_e^{y2} \hat{\mathbf{v}}_e \Big|_{\hat{\mathbf{v}}_e = \hat{\mathbf{v}}_e + \Delta \hat{\mathbf{v}}_e} \\
&= \sum_{i=0}^5 v_i^* \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^y (v_k - \Delta v_k) \right) (v_j - \Delta v_j) \\
&= \sum_{i=0}^5 v_i^* \left\{ \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^y v_k \right) v_j - \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^y v_k \right) \Delta v_j \right. \\
&\quad \left. - \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^y \Delta v_k \right) v_j + \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^y \Delta v_k \right) \Delta v_j \right\} \\
&= \sum_{i=0}^5 v_i^* \left\{ \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^y v_k \right) v_j - \sum_{j=0}^5 \left( \sum_{k=0}^5 \kappa_{ijk}^y v_k \right) \Delta v_j - \sum_{k=0}^5 \left( \sum_{j=0}^5 \kappa_{ijk}^y v_j \right) \Delta v_k \right\} \\
&= \sum_{i=0}^5 v_i^* \left\{ \sum_{j=0}^5 T_{ij}^{y2} v_j - \sum_{j=0}^5 T_{ij}^{y2} \Delta v_j - \sum_{k=0}^5 T_{ik}^{y1} \Delta v_k \right\} \\
&= \sum_{i=0}^5 v_i^* \sum_{j=0}^5 \{ T_{ij}^{y2} v_j - T_{ij}^{y2} \Delta v_j - T_{ij}^{y1} \Delta v_j \} \\
&= \hat{\mathbf{v}}_e^{*T} \hat{T}_e^{y2} \hat{\mathbf{v}}_e - \hat{\mathbf{v}}_e^{*T} \hat{T}_e^{y2} \Delta \hat{\mathbf{v}}_e - \hat{\mathbf{v}}_e^{*T} \hat{T}_e^{y1} \Delta \hat{\mathbf{v}}_e.
\end{aligned}$$

定常 Navier-Stokes 方程式に対する近似方程式は,

$$\begin{cases}
\nu \rho A \mathbf{u} - B \mathbf{p} - \rho D \mathbf{f} + \rho (S^{x2} \mathbf{u} + S^{y2} \mathbf{v}) = 0, \\
\nu \rho A \mathbf{v} - C \mathbf{p} - \rho D \mathbf{g} + \rho (T^{x2} \mathbf{u} + T^{y2} \mathbf{v}) = 0, \\
-B^T \mathbf{u} - C^T \mathbf{v} = 0.
\end{cases}$$

この近似方程式を解くことができれば, Navier-Stokes 方程式の数値解を得られる. しかし, 非線形であるので Stokes 方程式の様に連立 1 次方程式に帰着できない. そこで, Stokes 方程式の数値解を初期値として Newton 法を用いることで Navier-Stokes 方程式の数値解を求める.

Stokes 方程式の数値解を  $(\mathbf{u}^0, \mathbf{v}^0, \mathbf{p}^0)$  とする.

$$\begin{aligned}
\mathbf{u}^1 &= \mathbf{u}^0 - \Delta \mathbf{u}^0, \\
\mathbf{v}^1 &= \mathbf{v}^0 - \Delta \mathbf{v}^0, \\
\mathbf{p}^1 &= \mathbf{p}^0 - \Delta \mathbf{p}^0,
\end{aligned}$$

とおき,  $(\mathbf{u}^1, \mathbf{v}^1, \mathbf{p}^1)$  を近似方程式の  $(\mathbf{u}, \mathbf{v}, \mathbf{p})$  に代入する.

ただし,  $\Delta \mathbf{u}^0, \Delta \mathbf{v}^0, \Delta \mathbf{p}^0$  の 2 次の項は切り捨てる.

$$\begin{cases}
\nu \rho A (\mathbf{u}^0 - \Delta \mathbf{u}^0) - B (\mathbf{p}^0 - \Delta \mathbf{p}^0) - \rho D \mathbf{f} \\
\quad + \rho (S^{x2} \mathbf{u}^0 - S^{x2} \Delta \mathbf{u}^0 - S^{x1} \Delta \mathbf{u}^0) + \rho (S^{y2} \mathbf{v}^0 - S^{y2} \Delta \mathbf{v}^0 - T^{y1} \Delta \mathbf{u}^0) = 0, \\
\nu \rho A (\mathbf{v}^0 - \Delta \mathbf{v}^0) - C (\mathbf{p}^0 - \Delta \mathbf{p}^0) - \rho D \mathbf{g} \\
\quad + \rho (T^{x2} \mathbf{u}^0 - T^{x2} \Delta \mathbf{u}^0 - S^{x1} \Delta \mathbf{v}^0) + \rho (T^{y2} \mathbf{v}^0 - T^{y2} \Delta \mathbf{v}^0 - T^{y1} \Delta \mathbf{v}^0) = 0, \\
-(B^T \mathbf{u}^0 - B^T \Delta \mathbf{u}^0) - (C^T \mathbf{v}^0 - C^T \Delta \mathbf{v}^0) = 0.
\end{cases}$$

これを  $\Delta \mathbf{u}^0, \Delta \mathbf{v}^0, \Delta \mathbf{p}^0$  について整理する.

$$\left\{ \begin{array}{l} \rho S^{y2} \Delta \mathbf{v}^0 - B \Delta \mathbf{p}^0 \\ \quad + \rho(\nu A + S^{x1} + S^{x2} + T^{y1}) \Delta \mathbf{u}^0 = \rho(\nu A + S^{x2}) \mathbf{u}^0 + \rho S^{y2} \mathbf{v}^0 - B \mathbf{p}^0 - \rho D \mathbf{f}, \\ \rho T^{x2} \Delta \mathbf{u}^0 - C \Delta \mathbf{p}^0 \\ \quad + \rho(\nu A + S^{x1} + T^{y1} + T^{y2}) \Delta \mathbf{v}^0 = \rho T^{x2} \mathbf{u}^0 + \rho(\nu A + T^{y2}) \mathbf{v}^0 - C \mathbf{p}^0 - \rho D \mathbf{g}, \\ \quad -B^T \Delta \mathbf{u}^0 - C^T \Delta \mathbf{v}^0 = -B^T \mathbf{u}^0 - C^T \mathbf{v}^0. \end{array} \right.$$

上の式は  $(\Delta \mathbf{u}^0, \Delta \mathbf{v}^0, \Delta \mathbf{p}^0)$  に関する連立一次方程式になっている。  $\mathbf{u}^0, \mathbf{v}^0, \mathbf{p}^0$  は既知であるので右辺は既知量である。

よって、これを解くことで  $(\Delta \mathbf{u}^0, \Delta \mathbf{v}^0, \Delta \mathbf{p}^0)$  が求まり、第1ステップの近似値  $(\mathbf{u}^1, \mathbf{v}^1, \mathbf{p}^1)$  を得る。

第1ステップの近似値  $(\mathbf{u}^1, \mathbf{v}^1, \mathbf{p}^1)$  に対し

$$\begin{aligned} \mathbf{u}^2 &= \mathbf{u}^1 - \Delta \mathbf{u}^1, \\ \mathbf{v}^2 &= \mathbf{v}^1 - \Delta \mathbf{v}^1, \\ \mathbf{p}^2 &= \mathbf{p}^1 - \Delta \mathbf{p}^1, \end{aligned}$$

とおき、近似方程式に代入すれば、同様にして  $(\Delta \mathbf{u}^1, \Delta \mathbf{v}^1, \Delta \mathbf{p}^1)$  が求まるので、第2ステップ目の近似値  $(\mathbf{u}^2, \mathbf{v}^2, \mathbf{p}^2)$  を得る。以上の作業を繰り返すことにより、より厳密解に近い近似値が計算できる。(Newton法)

最初の近似値、すなわち Stokes 方程式の解より出発して、最終的には許容誤差内に入った時点で反復を打ち切り、その時の近似値を Navier-Stokes 方程式の数値解とする。

よって Navier-Stokes 方程式の数値解を求めることができたので、次節では具体例を挙げて数値実験を行う。

## 5 数値実験

### 5.1 実験の問題設定

Navier-Stokes 方程式

$$\begin{cases} -\Delta \mathbf{u} + (\mathbf{u} \cdot \nabla) \mathbf{u} + \nabla p = 0 & \text{in } \Omega, \\ -\operatorname{div} \mathbf{u} = 0 & \text{in } \Omega, \\ \mathbf{u} = \mathbf{b} & \text{on } \partial\Omega. \end{cases}$$

について考える. ( $\nu = \rho = 1, \mathbf{f} = 0$  とした)

領域  $\Omega$  を以下のような円環領域とする.

$$\Omega = \{\mathbf{x} \in \mathbb{R}^2; a < |\mathbf{x}| < 1\}.$$

ただし  $a$  は  $0 < a < 1$  をみたす定数である. また,  $\Gamma_1, \Gamma_2$  を次のようにおくと,

$$\Gamma_1 = \{\mathbf{x} \in \mathbb{R}^2; |\mathbf{x}| = a\},$$

$$\Gamma_2 = \{\mathbf{x} \in \mathbb{R}^2; |\mathbf{x}| = 1\},$$

境界  $\partial\Omega$  は  $\partial\Omega = \Gamma_1 \cup \Gamma_2$  となる.

また境界値,  $\mathbf{b}(x, y)$  は Gauss の発散定理

$$\iint_{\Omega} \operatorname{div} \mathbf{u} \, dx \, dy = \int_{\partial\Omega} \mathbf{b} \cdot \mathbf{n} \, dS,$$

により, 条件

$$\sum_{i=1}^2 \int_{\Gamma_i} \mathbf{b} \cdot \mathbf{n} \, dS = 0.$$

をみたさなければならない. この条件のことを一般流束条件という.

各  $i$  ( $i = 1, 2$ ) について

$$\int_{\Gamma_i} \mathbf{b} \cdot \mathbf{n} \, dS = 0$$

をみたすならば, 一般流束条件をみたす. このときは Navier-Stokes 方程式の解の存在が解析的に示されている. (H.Fujita, O.A.Ladyzhenskaya)

しかし, この条件が成立せず, 一般流束条件のみが成立する場合は, 解が一般的にあるかどうかは知られていない. そこで, 境界値が特殊な場合 2 通りについて数値実験を行う.

#### 5.1.1 数値実験 1

Navier-Stokes 方程式と領域は上の通りで, 境界値を次のようにする.

$$\mathbf{b} = \frac{\mu}{R_i} \mathbf{e}_r + R_i \omega_i \mathbf{e}_\theta \quad \text{on } \Gamma_i \quad (\mu \neq 0, i = 1, 2),$$

ただし,  $\mathbf{e}_r, \mathbf{e}_\theta$  は動径方向, 偏角方向の単位ベクトル,  $\mu, \omega_1, \omega_2$  は与えられた定数,  $R_1 = a, R_2 = 1$  とする.

このとき以下のような厳密解  $u_0$  が存在することが知られている. (H.Morimoto[7])

$$u_0 = \frac{\mu}{r} \mathbf{e}_r + b(\mu, r) \mathbf{e}_\theta.$$

ここで,  $r$  は各点の原点からの距離で,  $b(\mu, r)$  は  $\mu$  の値によって以下のように変わる.

(1)  $\mu \neq -2\nu$  のとき,

$$b(\mu, r) = \frac{c_1}{r} + c_2 r^{1+\frac{\mu}{\nu}},$$

$$c_1 = \frac{\omega_1 a^2 - \omega_2 a^{2+\frac{\mu}{\nu}}}{1 - a^{2+\frac{\mu}{\nu}}}, \quad c_2 = \frac{\omega_2 - \omega_1 a^2}{1 - a^{2+\frac{\mu}{\nu}}}.$$

(2)  $\mu = -2\nu$  のとき,

$$b(\mu, r) = \frac{1}{r} (c_1 + c_2 \log r),$$

$$c_1 = \omega_2, \quad c_2 = \frac{\omega_1 a^2 - \omega_2}{\log a}.$$

また, このようにした  $\mathbf{b}$  は一般流束条件のみをみたす.

なぜなら, 外向き単位法線ベクトル  $\mathbf{n}$  が

$$\mathbf{n} = \begin{cases} -\mathbf{e}_r & \text{on } \Gamma_1 \\ \mathbf{e}_r & \text{on } \Gamma_2 \end{cases}$$

なので,

$$\begin{aligned} \int_{\Gamma_1} \mathbf{b} \cdot \mathbf{n} dS &= \int_{\Gamma_1} \left( \frac{\mu}{a} \mathbf{e}_r + a\omega_i \mathbf{e}_\theta \right) \cdot (-\mathbf{e}_r) dS \\ &= - \int_{\Gamma_1} \frac{\mu}{a} dS = - \int_0^{2\pi} \frac{\mu}{a} a d\theta \\ &= -2\pi\mu, \end{aligned}$$

$$\begin{aligned} \int_{\Gamma_2} \mathbf{b} \cdot \mathbf{n} dS &= \int_{\Gamma_2} (\mu \mathbf{e}_r + \omega_i \mathbf{e}_\theta) \cdot \mathbf{e}_r dS \\ &= \int_{\Gamma_2} \mu dS = \int_0^{2\pi} \mu d\theta \\ &= 2\pi\mu. \end{aligned}$$

よって,  $\mu \neq 0$  ならば, 一般流束条件のみをみたす.

実験 1 としては,  $a = 0.5$  を固定して, パラメータ  $\mu, \omega_1, \omega_2$  を動かして考える. このとき, 数値解の様子を確認するとともに, 有限要素法における要素分割数と, 数値解と厳密解との誤差の関係を図示してみる.

## 5.1.2 数値実験 2

今度は, 実験 1 で考えた境界値に摂動  $\varphi_i(\theta)$ ,  $\psi_i(\theta)$  を加えた場合を考える.

$$\mathbf{b} = \left( \frac{\mu}{R_i} + \varphi_i(\theta) \right) \mathbf{e}_r + (R_i \omega_i + \psi_i(\theta)) \mathbf{e}_\theta \quad \text{on } \Gamma_i \quad (i = 1, 2),$$

ただし,  $\varphi_i(\theta)$ ,  $\psi_i(\theta)$  は周期  $2\pi$  の滑らかな関数で, 以下の条件をみたす.

$$\int_0^{2\pi} \varphi_i(\theta) d\theta = 0, \quad \int_0^{2\pi} \psi_i(\theta) d\theta = 0$$

摂動を加えた場合も, 同様に  $\mu \neq 0$  ならば一般流束条件のみをみたす.

このとき,  $\nu, a, \omega_1, \omega_2$  が以下の条件

$$|\omega_1 - \omega_2| \frac{a^2}{1 - a^2} (\log a)^2 < 2\nu$$

をみたしている場合, ある 1 次元可算集合  $M$  が存在して, 任意の  $\mu \in \mathbb{R} \setminus M$  に対し,  $\varphi_i$ ,  $\psi_i$  ( $i = 1, 2$ ) が十分小さいならば Navier-Stokes 方程式の解が存在することが示されている. (H.Morimoto and S.Ukai[8])

実験 2 としては,

$$a = 0.5, \quad (\varphi_i(\theta), \psi_i(\theta)) = (AR_i \cos \theta, BR_i \sin \theta) \quad (i = 1, 2, \quad A \text{ と } B \text{ は任意の実数})$$

と固定して考える. このとき, パラメータ  $\mu, \omega_1, \omega_2, A, B$  を動かした場合の数値解の存在と様子を調べる. また, 条件式を  $\omega_1, \omega_2$  について整理すると

$$|\omega_1 - \omega_2| < \frac{6}{(\log 2)^2} \doteq 12.488$$

となるので, この式をみたす範囲とみたさない範囲両方について調べる.

## 5.2 実験結果

### 5.2.1 数値実験 1 の結果

$$\mathbf{b} = \frac{\mu}{R_i} \mathbf{e}_r + R_i \omega_i \mathbf{e}_\theta \quad \text{on } \Gamma_i \quad (i = 1, 2),$$

境界値  $\mathbf{b}$  が上記のとき,

[case 1-1]  $\mu = \omega_1 = \omega_2 = 1$

[case 1-2]  $\mu = 10, \omega_1 = \omega_2 = 1$

[case 1-3]  $\mu = 1, \omega_1 = 3, \omega_2 = 7$

[case 1-4]  $\mu = 1, \omega_1 = -10, \omega_2 = 10$

の 4 つの場合について, 流速のベクトル場と等圧線, 加えて厳密解と数値解の誤差と, 要素分割数との関係について図示してみる.

ただし, 流速については見やすくするために長さを  $d$  倍に調整し, 圧力については高低がわかるように色をつけた. (倍率  $d$  は各グラフに併記, 圧力の高低は 青 < 水 < 緑 < 深紅 < 赤, 実線 < 点線 とした.)



[case 1-1] ( $\mu = \omega_1 = \omega_2 = 1$ ) の場合

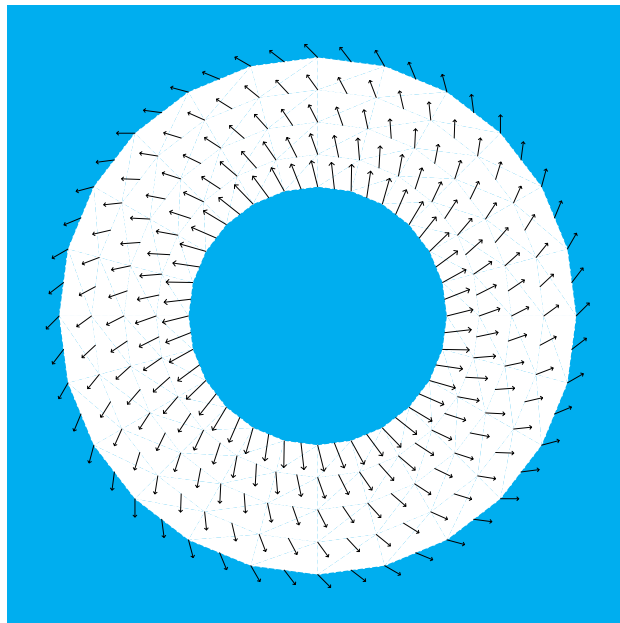


図 1: 数値解の流速 ( $d = 0.05$ )

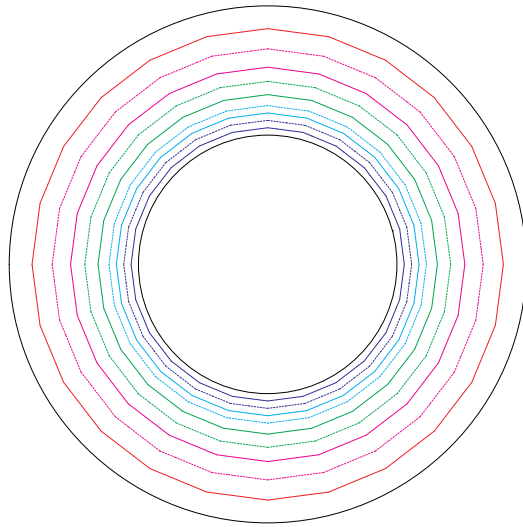


図 2: 数値解の等圧線

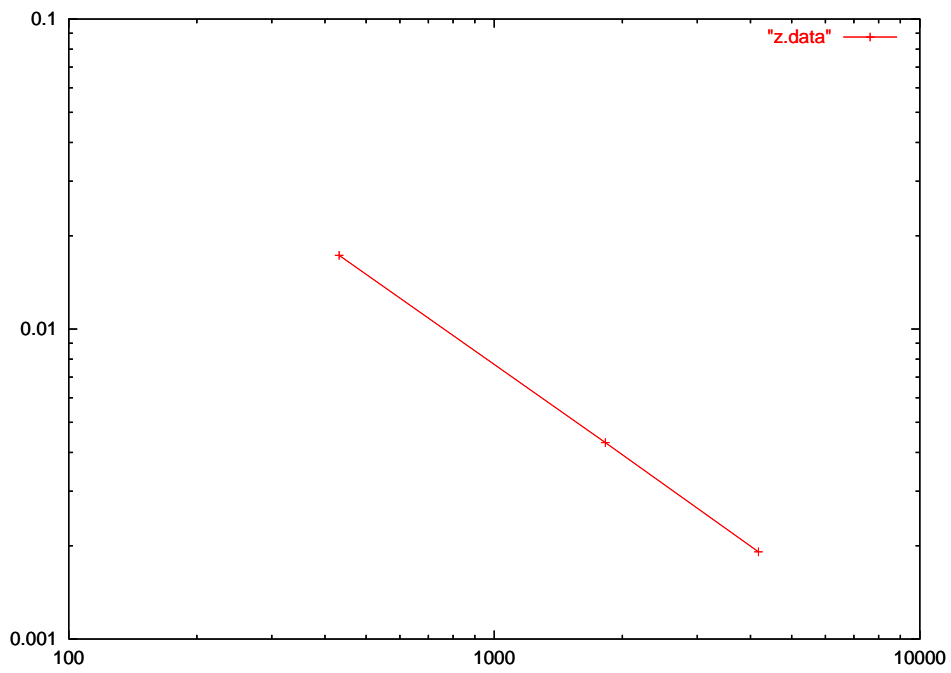


図 3: 要素分割数 (横軸) と誤差 (縦軸) を両側対数グラフ上にプロット

[case 1-2] ( $\mu = 10$  ,  $\omega_1 = \omega_2 = 1$ ) の場合

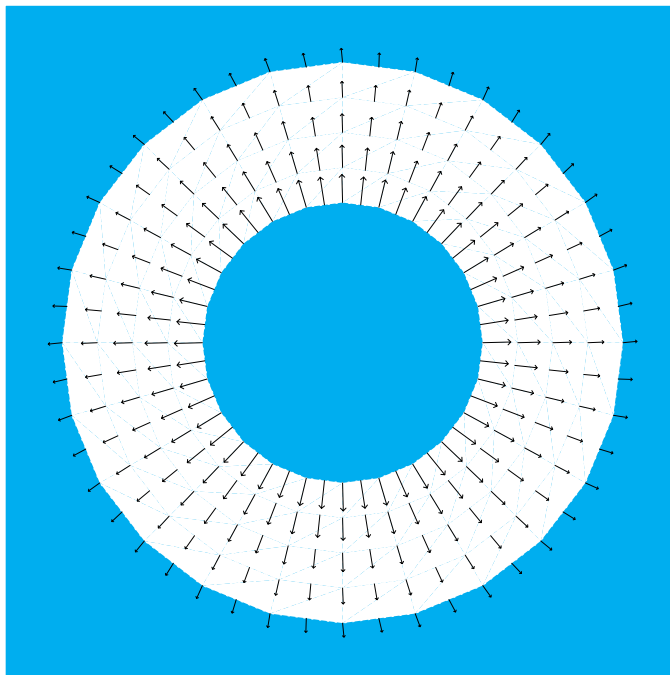


図 4: 数値解の流速 ( $d = 0.005$ )

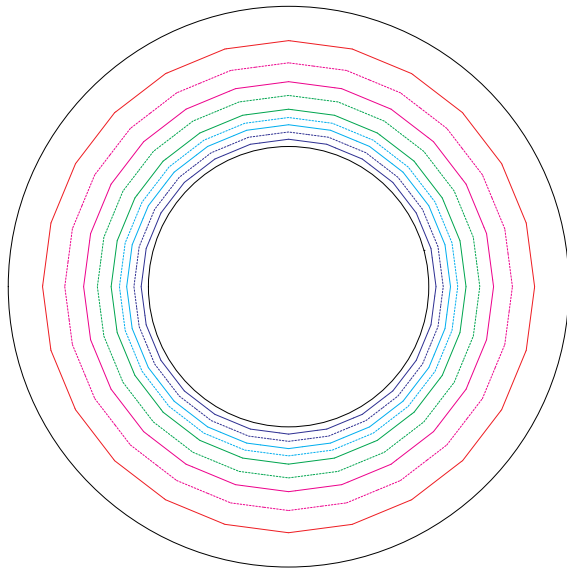


図 5: 数値解の等圧線

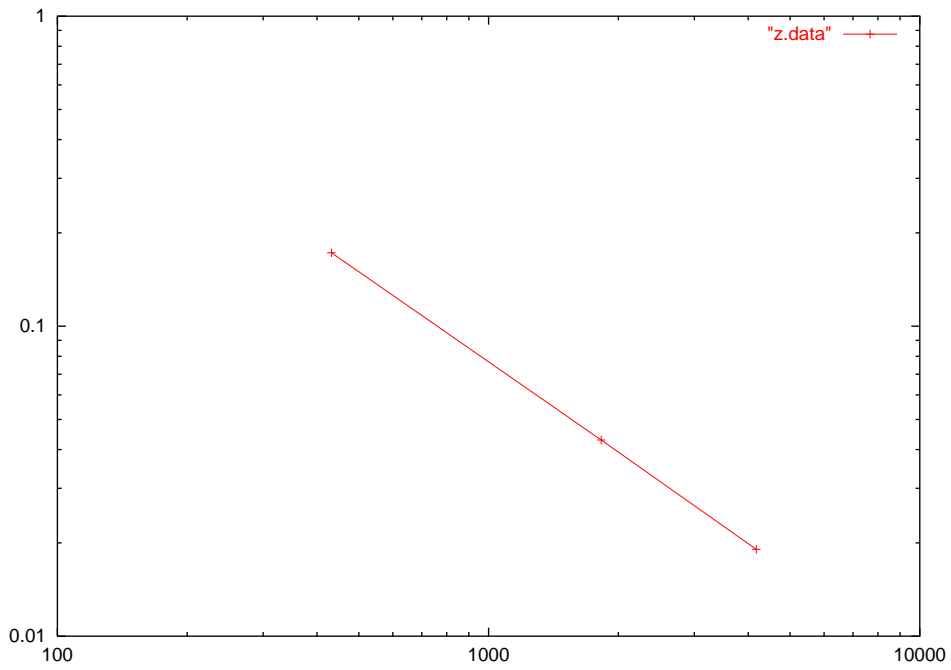


図 6: 要素分割数 (横軸) と誤差 (縦軸) を両側対数グラフ上にプロット

[case 1-3] ( $\mu = 1$  ,  $\omega_1 = 3$  ,  $\omega_2 = 7$ ) の場合

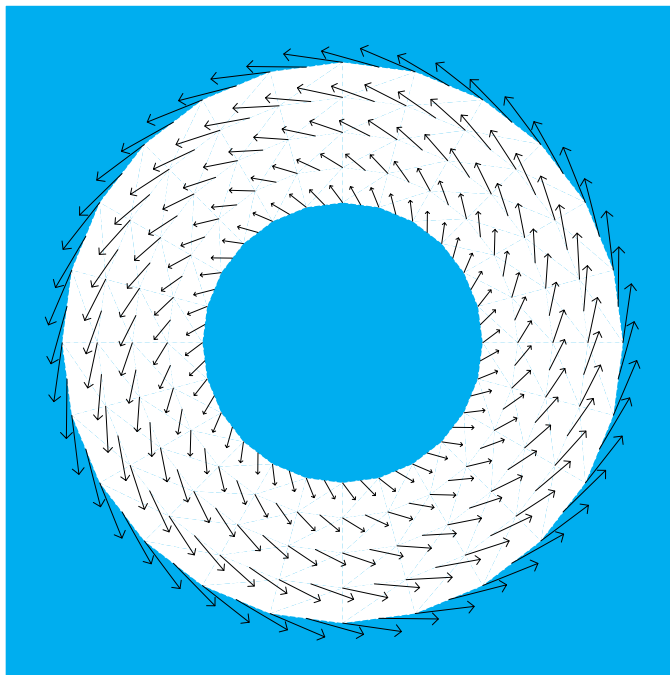


図 7: 数値解の流速 ( $d = 0.03$ )

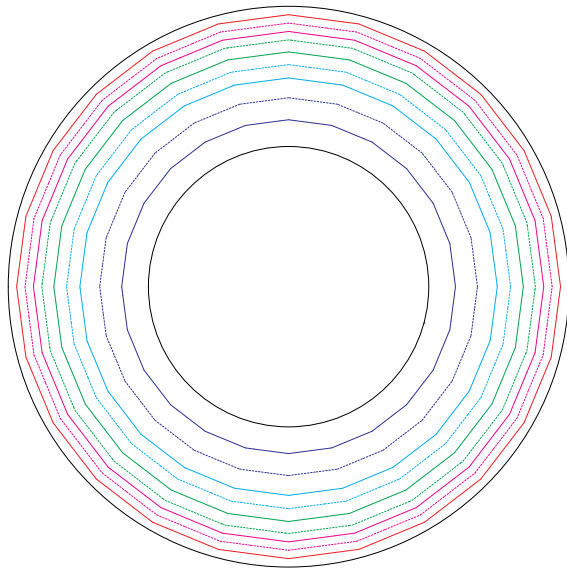


図 8: 数値解の等圧線

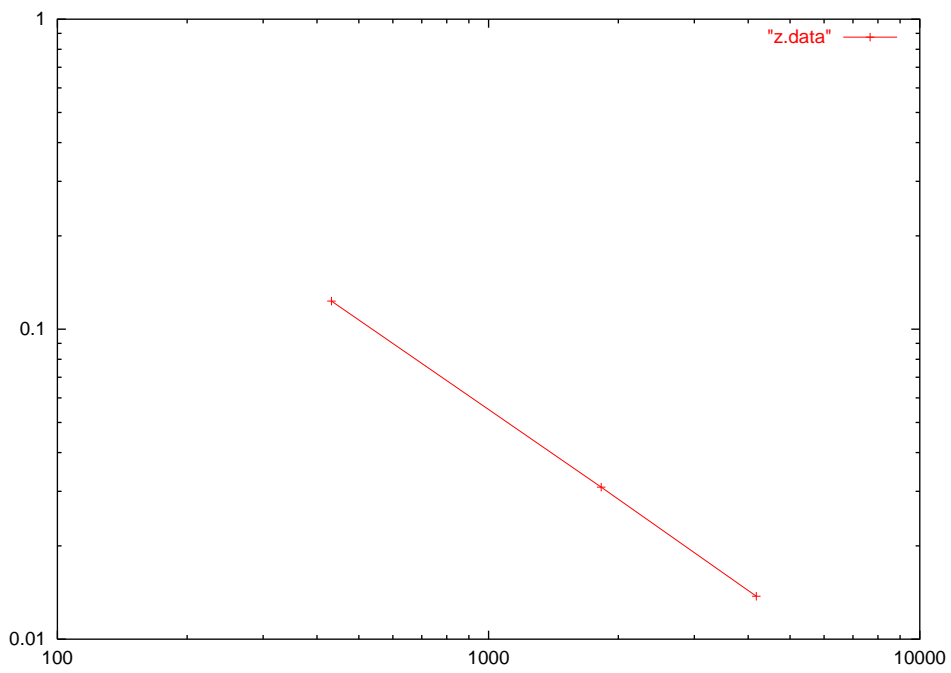


図 9: 要素分割数 (横軸) と誤差 (縦軸) を両側対数グラフ上にプロット

[case 1-4] ( $\mu = 1$  ,  $\omega_1 = -10$  ,  $\omega_2 = 10$ ) の場合

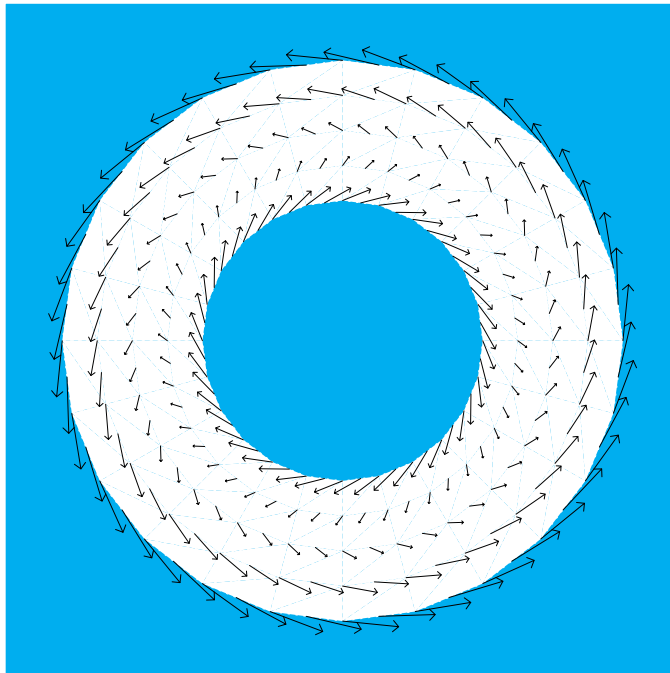


図 10: 数値解の流速 ( $d = 0.02$ )

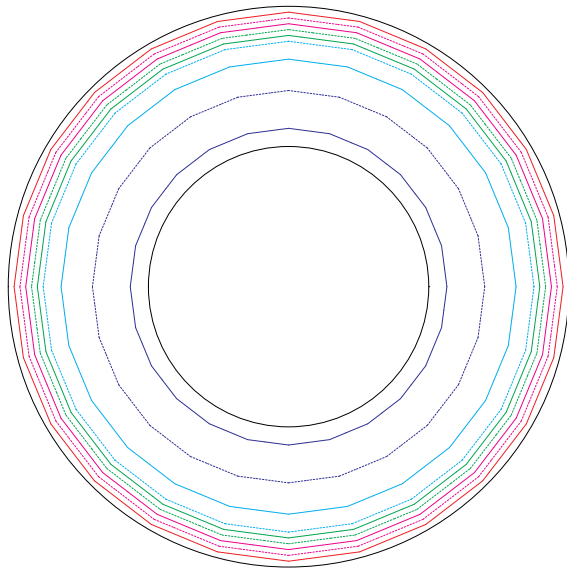


図 11: 数値解の等圧線

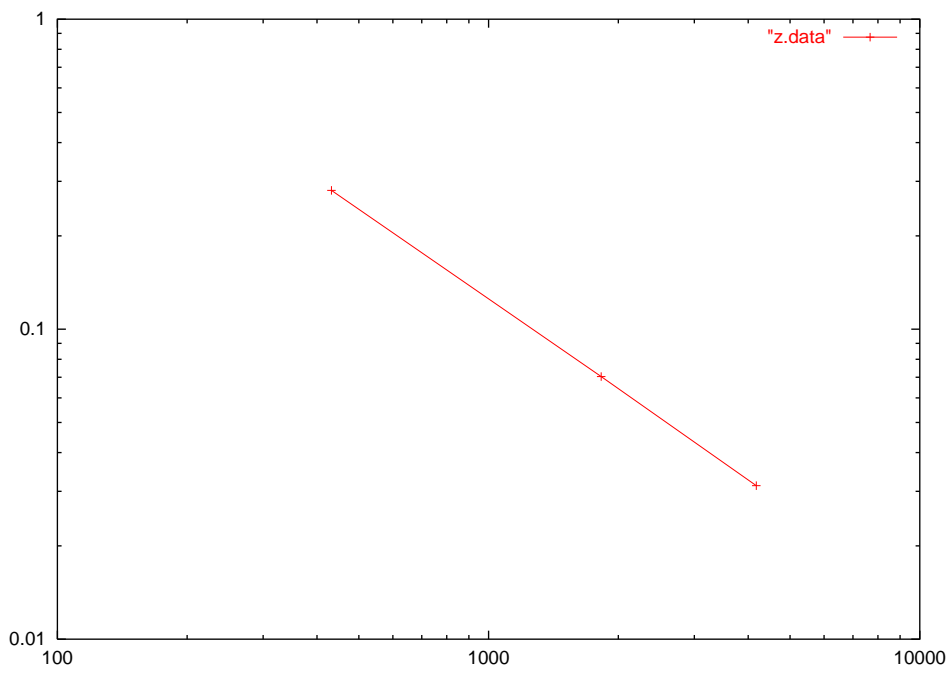


図 12: 要素分割数 (横軸) と誤差 (縦軸) を両側対数グラフ上にプロット



### 5.3 数値実験 2 の結果

$$\mathbf{b} = \left( \frac{\mu}{R_i} + AR_i \cos \theta \right) \mathbf{e}_r \\ + (R_i \omega_i + BR_i \sin \theta) \mathbf{e}_\theta \quad \text{on } \Gamma_i \quad (i = 1, 2),$$

境界値  $\mathbf{b}$  が上記のとき,

[case 2-1]  $\mu = \omega_1 = \omega_2 = 1$

[case 2-2]  $\mu = 10, \omega_1 = \omega_2 = 1$

[case 2-3]  $\mu = 1, \omega_1 = 3, \omega_2 = 7$

[case 2-4]  $\mu = 1, \omega_1 = -10, \omega_2 = 10$

の 4 つの場合について, 摂動の係数  $A, B$  を  $A = B = 0.01, 0.1, 1, 10$  としたときの流速のベクトル場と等圧線を描画する.

このとき, [case 1-1], [case 1-2], [case 1-3] は解の存在条件をみたしているが, [case 1-4] は条件をみたしていない.

また実験 1 と同様に, 流速については見やすくするために長さを  $d$  倍に調整し, 圧力については高低がわかるように色をつけた.

[case 2-1] において  $A = B = 0.01$  の場合

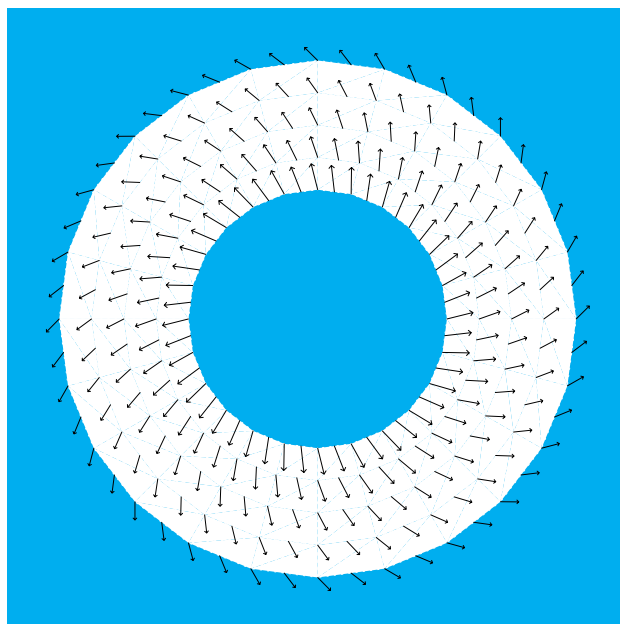


図 13: 流速 ( $d = 0.05$ )

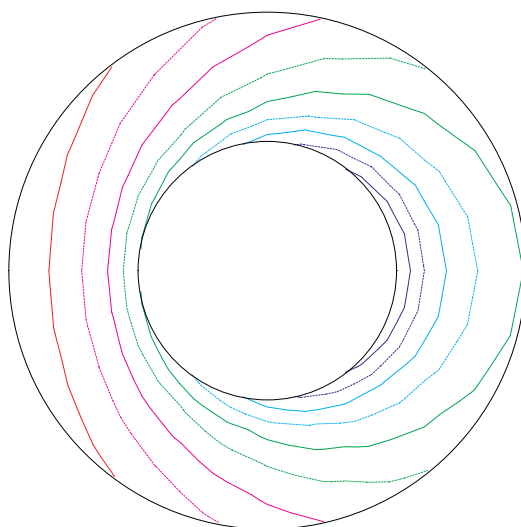


図 14: 等圧線

[case 2-1] において  $A = B = 0.1$  の場合

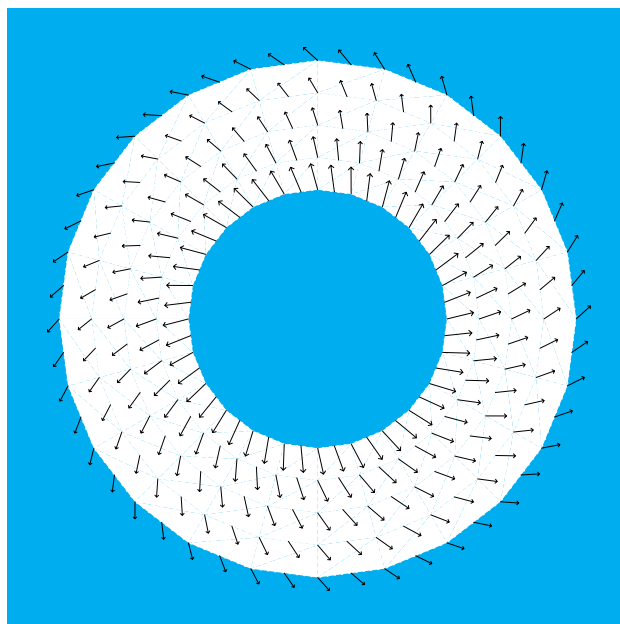


図 15: 流速 ( $d = 0.05$ )

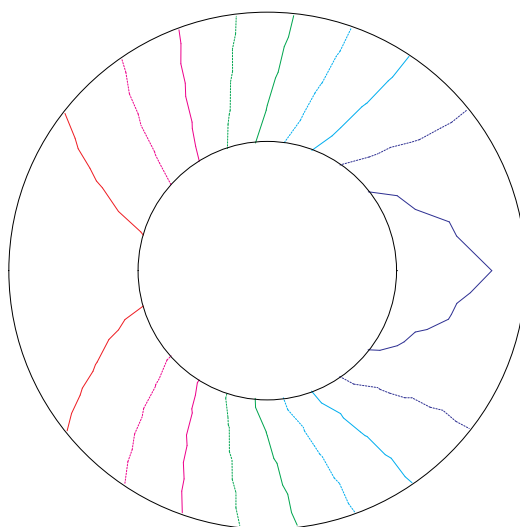


図 16: 等圧線

[case 2-1] において  $A = B = 1$  の場合

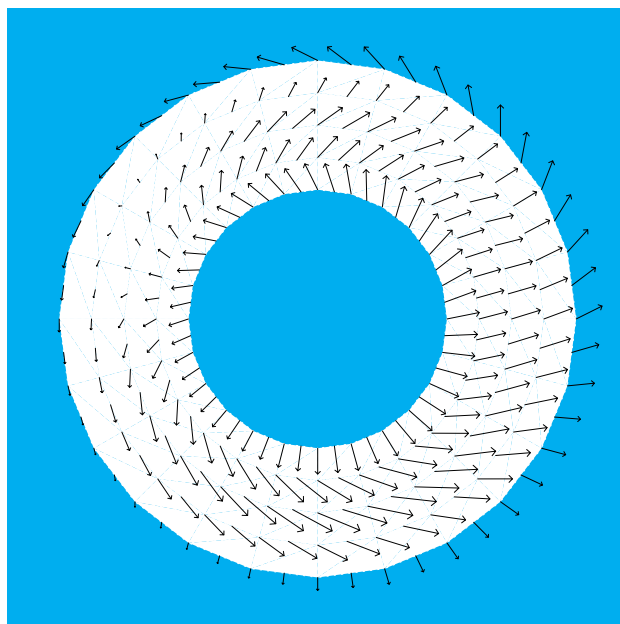


図 17: 流速 ( $d = 0.05$ )

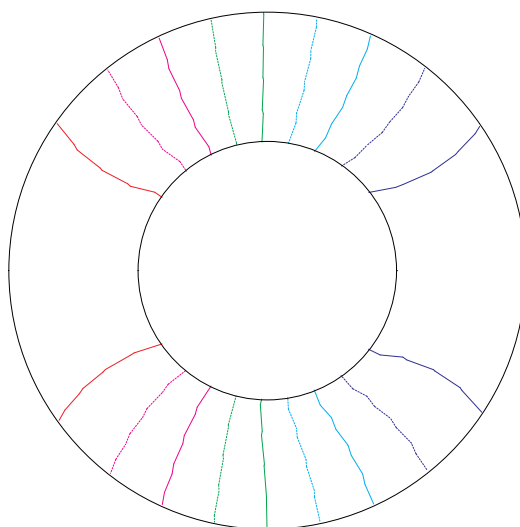


図 18: 等圧線

[case 2-1] において  $A = B = 10$  の場合

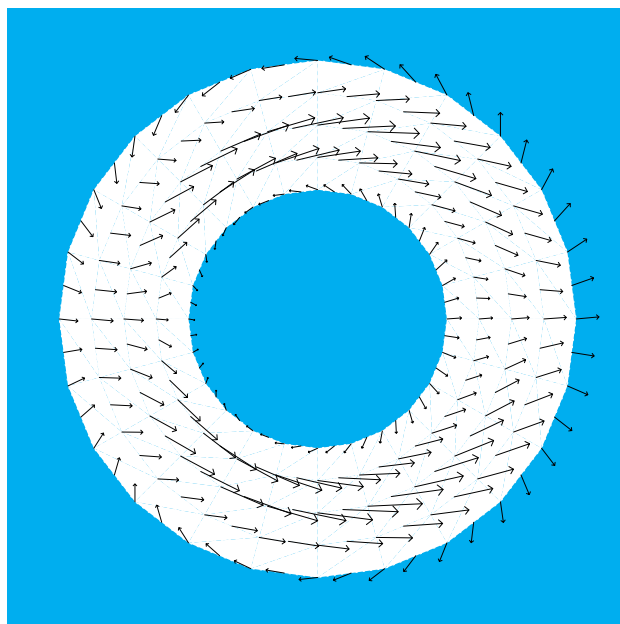


図 19: 流速 ( $d = 0.008$ )

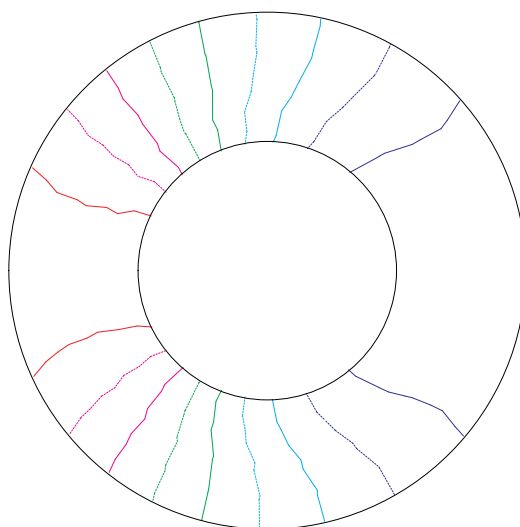


図 20: 等圧線

[case 2-2] において  $A = B = 0.01$  の場合

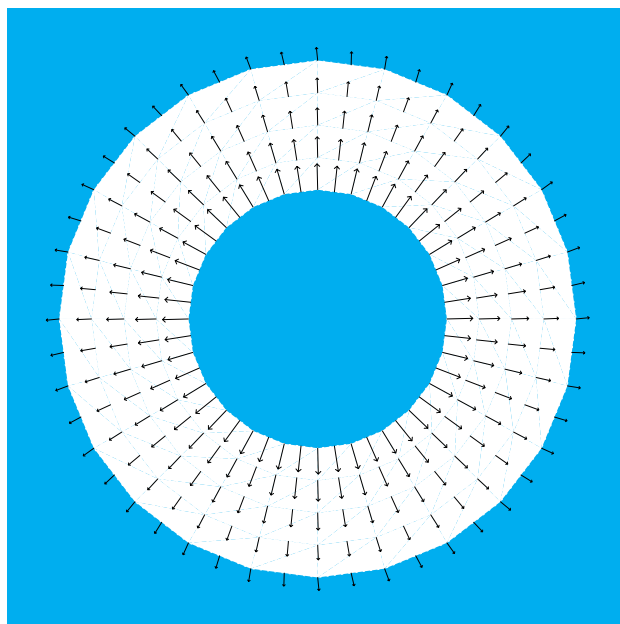


図 21: 流速 ( $d = 0.005$ )

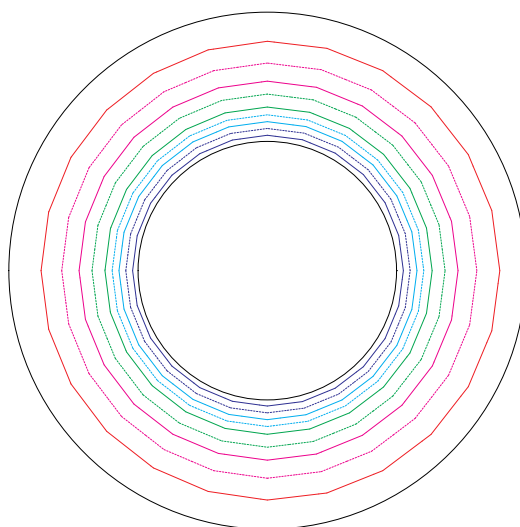


図 22: 等圧線

[case 2-2] において  $A = B = 0.1$  の場合

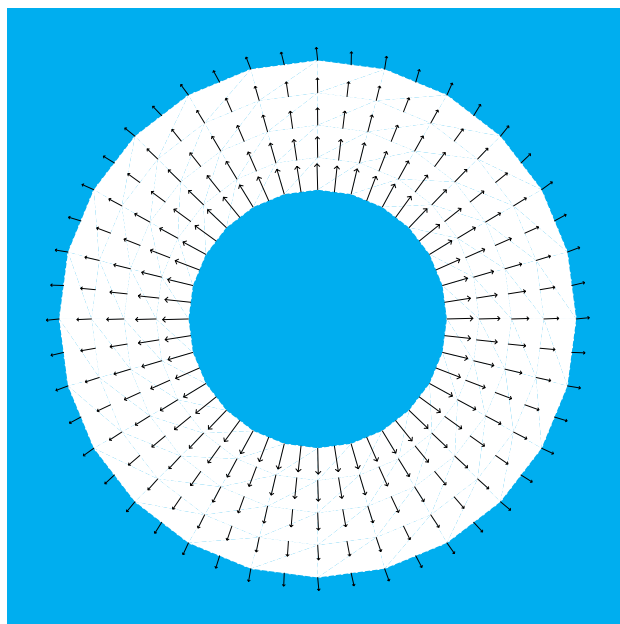


図 23: 流速 ( $d = 0.005$ )

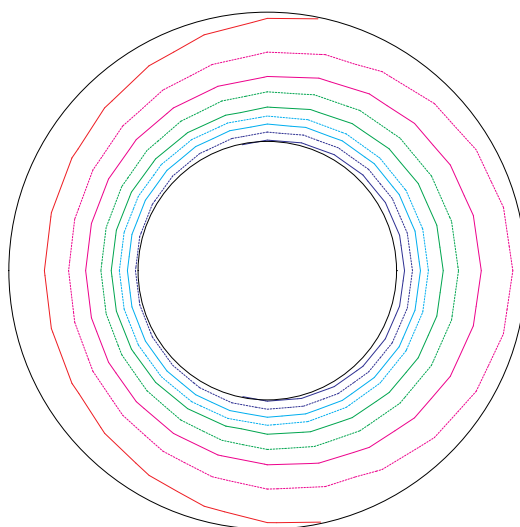


図 24: 等圧線

[case 2-2] において  $A = B = 1$  の場合

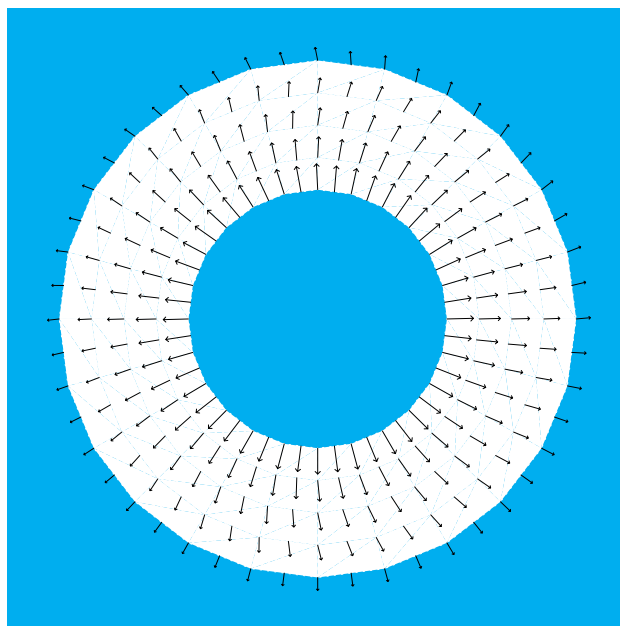


図 25: 流速 ( $d = 0.005$ )

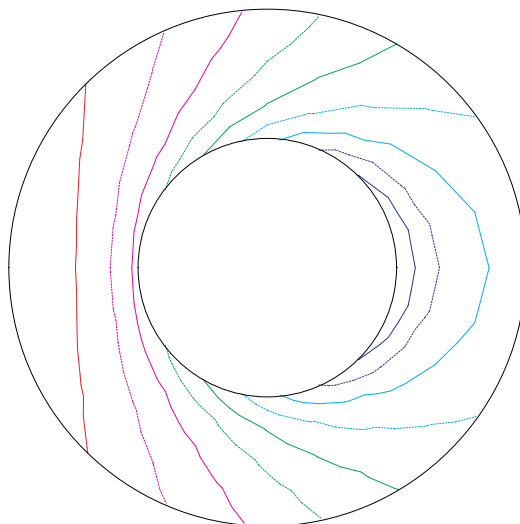


図 26: 等圧線



[case 2-2] において  $A = B = 10$  の場合

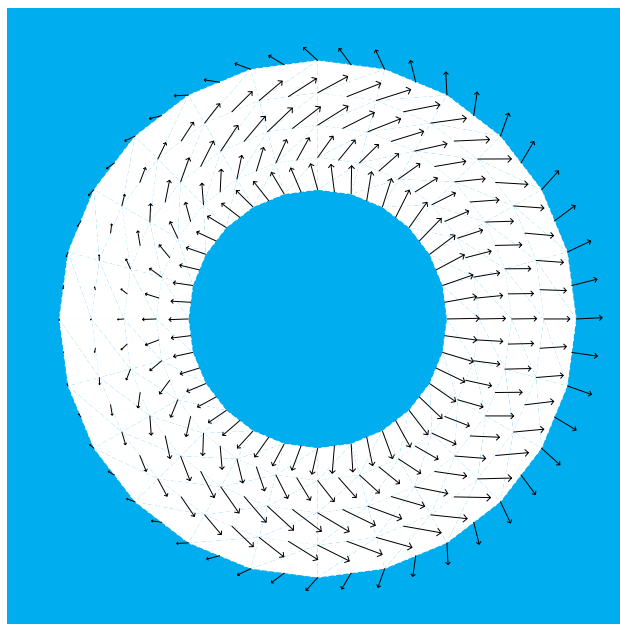


図 27: 流速 ( $d = 0.005$ )

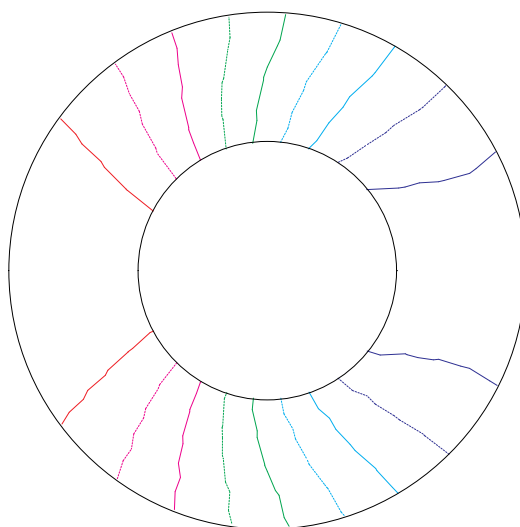


図 28: 等圧線

[case 2-3] において  $A = B = 0.01$  の場合

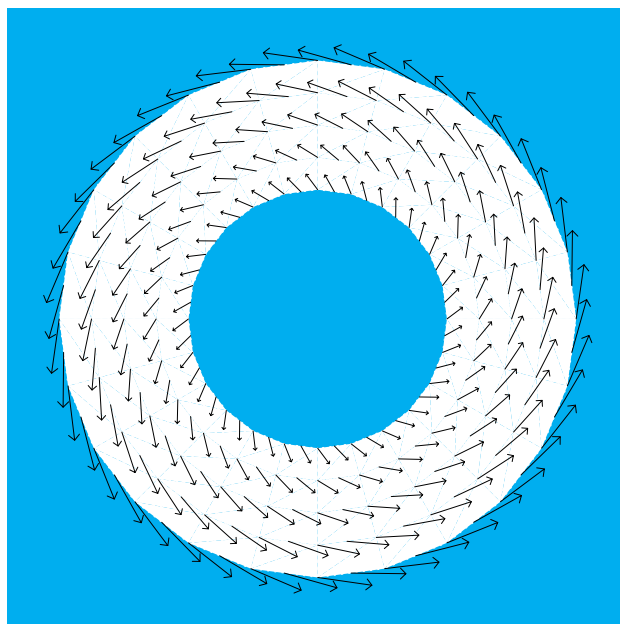


図 29: 流速 ( $d = 0.03$ )

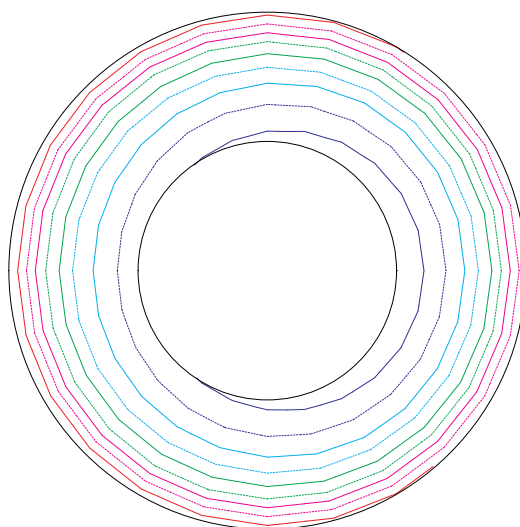


図 30: 等圧線

[case 2-3] において  $A = B = 0.1$  の場合

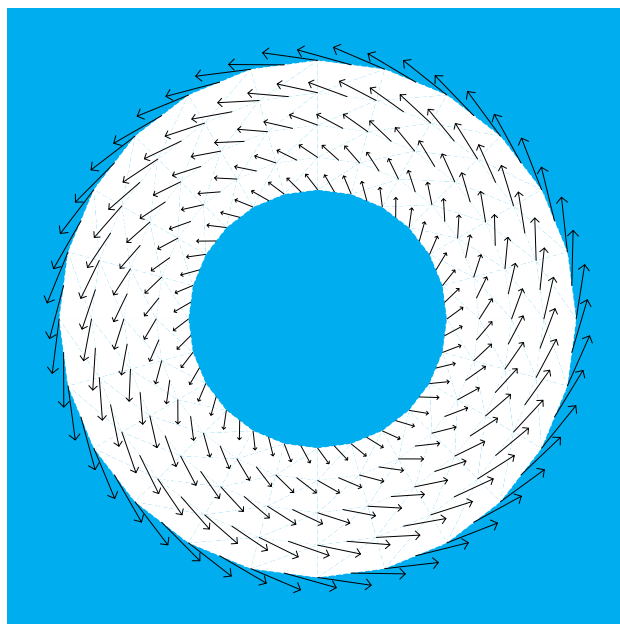


図 31: 流速 ( $d = 0.03$ )

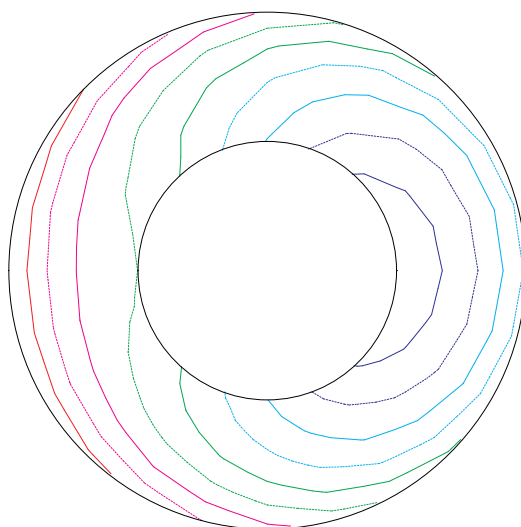


図 32: 等圧線

[case 2-3] において  $A = B = 1$  の場合

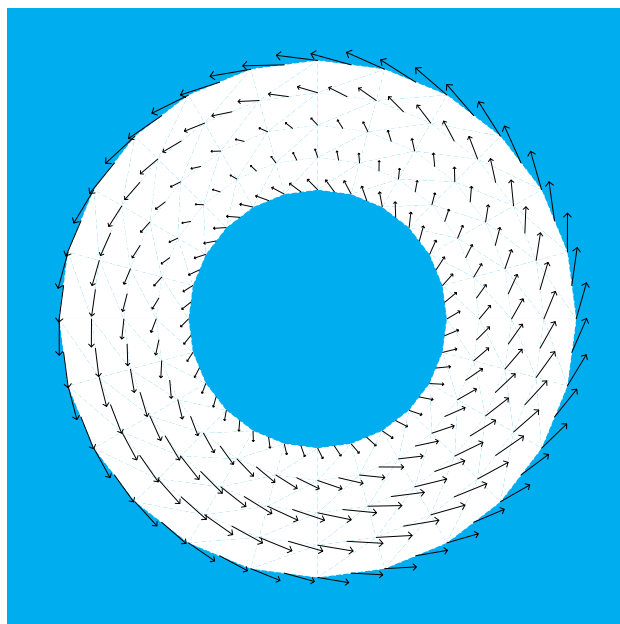


図 33: 流速 ( $d = 0.02$ )

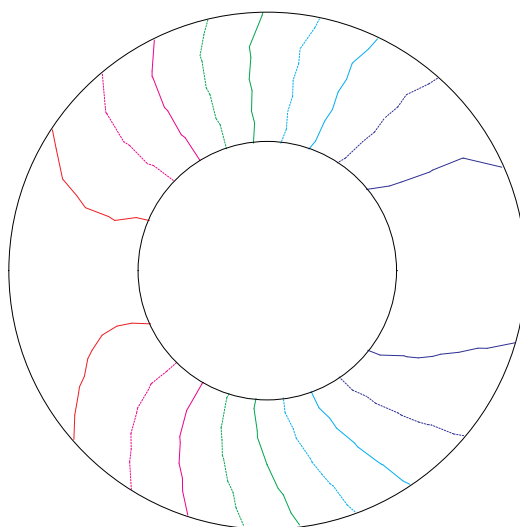


図 34: 等圧線

[case 2-3] において  $A = B = 10$  の場合

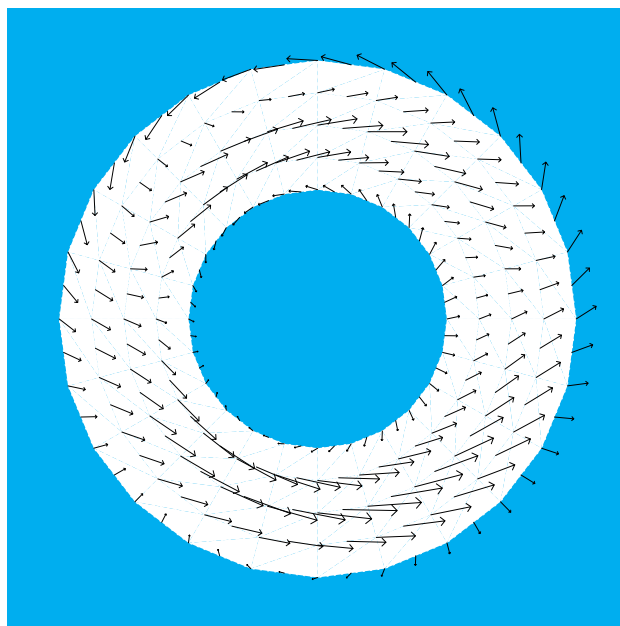


図 35: 流速 ( $d = 0.007$ )

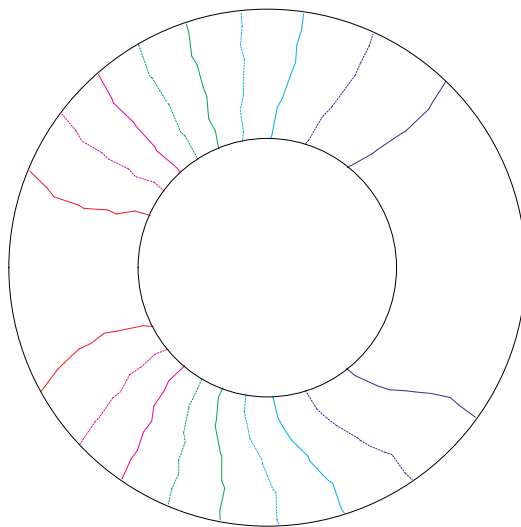


図 36: 等圧線

[case 2-4] において  $A = B = 0.01$  の場合

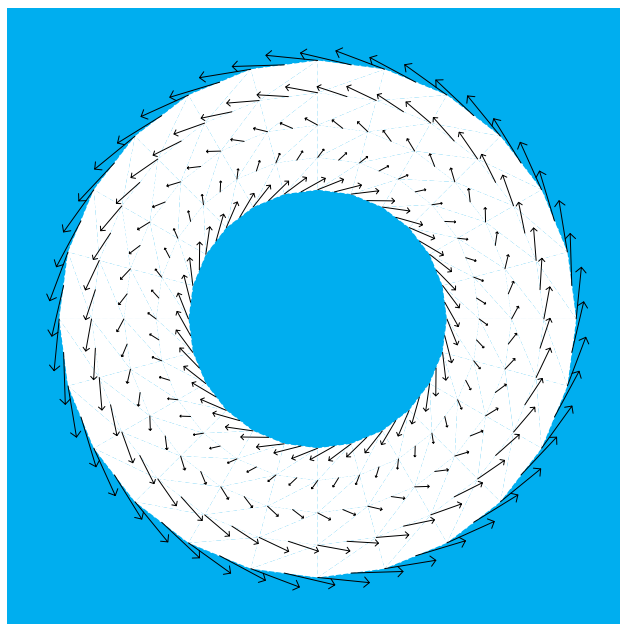


図 37: 流速 ( $d = 0.02$ )

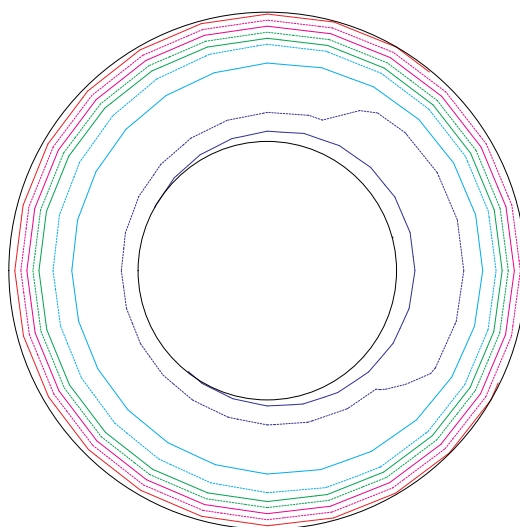


図 38: 等圧線

[case 2-4] において  $A = B = 0.1$  の場合

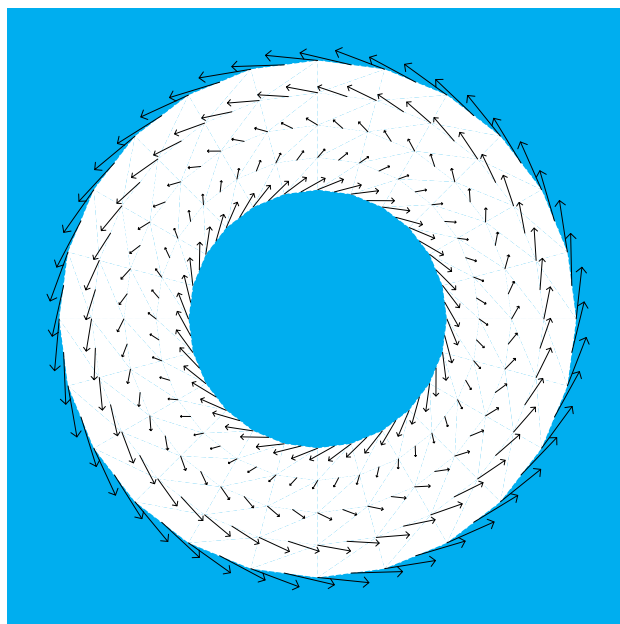


図 39: 流速 ( $d = 0.02$ )

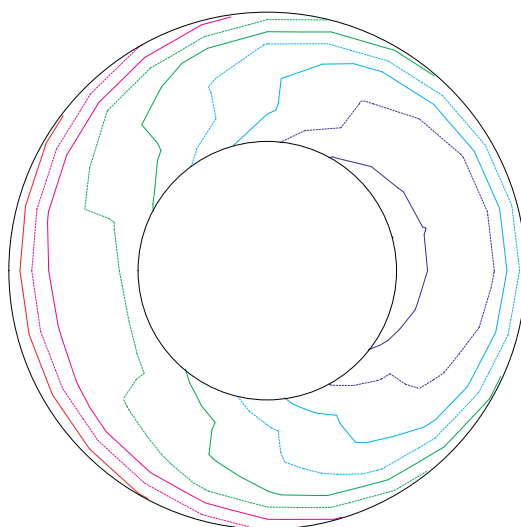


図 40: 等圧線

[case 2-4] において  $A = B = 1$  の場合

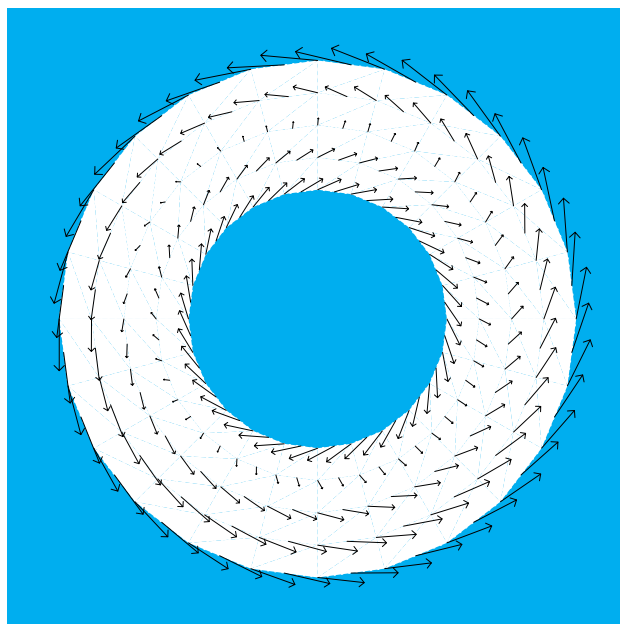


図 41: 流速 ( $d = 0.02$ )

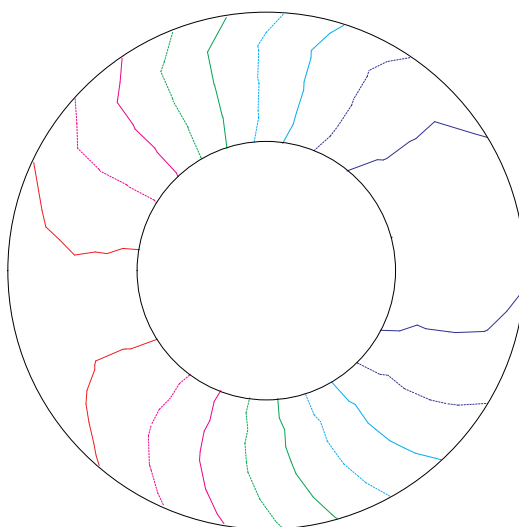


図 42: 等圧線



[case 2-4] において  $A = B = 10$  の場合

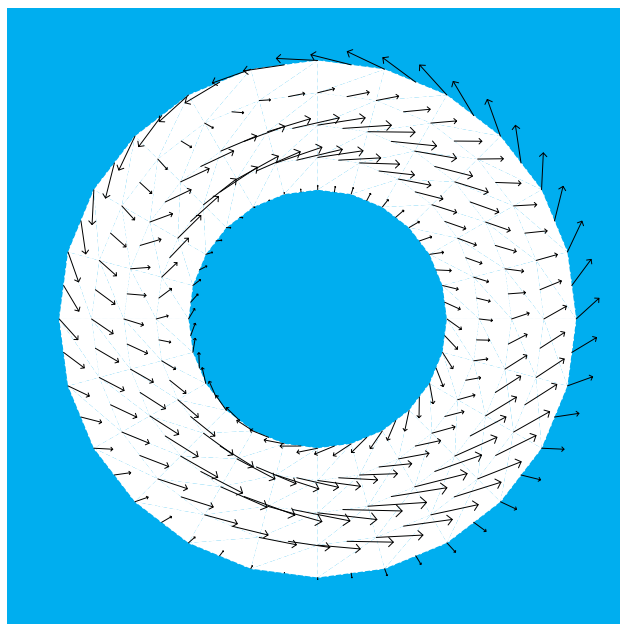


図 43: 流速 ( $d = 0.008$ )

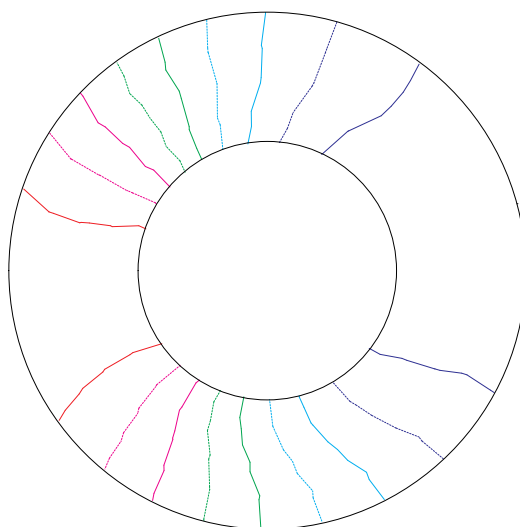


図 44: 等圧線

## 5.4 結果のまとめ

実験1について, どの場合でも厳密解と数値解の誤差と, 要素分割数を対数グラフ上にプロットすると直線状に分布していた. その傾きが大体  $-1/2$  であることから, 要素分割数が大きくなると誤差は  $-1/2$  のオーダーで減少していく. すなわち, 要素の直径  $h$  が小さくなると, 誤差は  $h^2$  のオーダーで減少していくといえる.

実験2について, どの場合でも数値解が存在しているので, 解の存在条件をみたさないところでも解が存在すると推察できる. また, 一部の例で摂動が大きいところで流れが止まる点が出現したり, 同心円状に分布していた圧力が左右に分かれる現象が発生したり興味深い結果を得ることができた.

## 6 実験で用いたプログラム

本節では, 数値計算を行うために使用したプログラムを紹介する.

- [1] 領域分割と境界値作成を行うプログラム `annulus.c`
- [2] Stokes 方程式を解くプログラム `stokes-fukushima.c`
- [3] 行列  $\kappa_{ijk}^x, \kappa_{ijk}^y$  の計算プログラム (Mathematica による)
- [4] Navier-Stokes 方程式を解くプログラム `navier3-fukushima.c`
- [5] 流速のベクトル場を描くプログラム `fluid3.c`
- [6] 等圧線を描くプログラム `contour.c`

[3] は自作, 他のプログラムは全て工藤丈征氏のプログラムが元になっていて, それを桂田 祐史助教が改良したのをさらに改良した.

### 6.1 領域分割と境界値作成を行うプログラム

```
1 /*
2  * annulus.c --- 円環領域で N-S を解くための入力データを作る。
3  *             元は File/file.c (工藤氏作成, 桂田先生改良)
4  *             さらに, 分割数, 摂動を含めた境界値のデータ
5  *             などが入力できるように改良 (福嶋)
6  *
7  * たとえば, マニュアルの例題を解くための file.data を作るには
8  *
9  *             ccmg annulus.c
10 *             ./annulus file.data
11 *             (v, rho = 1 1 に固定)
12 *             (func forc = 0 0 に固定)
13 *
14 * init() の factor を 4 にして 2.477431e-03
15 *             2           3.549791e-02
16 *             1           4.686050e-01
17 */
18
19 #include <math.h>
20 #include <stdio.h>
21 #include "matutil.h"
22
23 int M; /* M 5 */           /* 動径方向の分割 */
```

```

24 int N; /* N 24 */ /* 偏角方向の分割 */
25 int U; /* 432 */ /* 流速の節点の数 U = 2 N (2 M - 1) */
26 int P; /* 120 */ /* 圧力の節点の数 P = M × N */
27 int T; /* 192 */ /* 三角形の数 T = 2 N (M - 1) */
28 int SIZE; /* 984 */ /* 未知数の数 S I Z E = N (9 M - 4) */
29
30 int M1; /* 4 */ /* M - 1 */
31 int N2; /* 48 */ /* 2 N */
32 int N4; /* 96 */ /* 4 N */
33 int N9; /* 216 */ /* 9 N */
34 int U2; /* 864 */ /* 2 U */
35
36 double R1; /* 2.0 */ /* 内側の半径の値 */
37 double R2; /* 5.0 */ /* 外側の半径の値 */
38 double R; /* 5.3 */ /* R = R 2 + α fspase(-R,-R,R,R) */
39
40 void init()
41 {
42     printf("M 動径方向の分割数の入力\n");
43     scanf("%d", &M);
44     printf("N 偏角方向の分割数の入力\n");
45     scanf("%d", &N);
46     printf("R1 内側の円の半径\n");
47     scanf("%lf", &R1);
48     U=2*N*(2*M-1); P = M * N; T = 2 * N * (M - 1);
49     SIZE = N*(9*M-4);
50     M1 = M - 1; N2 = 2 * N; N4 = 4 * N; N9 = 9 * N; U2 = 2 * U;
51     R2 = 1.0; R = R2 * (1.3/1);
52     printf("M=%d, N=%d, 節点の個数=%d, 未知数の個数=%d\n", M, N, U, SIZE);
53 }
54
55 #define C 3
56 /*
57 * C = 1 ならば内側で細かく分割
58 * C = 2 ならば外側で細かく分割
59 * 上以外ならば等間隔に分割する
60 */
61
62 void number_p(int k, int z[3])
63 {
64     int h;
65     h = k / N2;
66
67     if (k % N2 < N) {
68         z[0] = k - h * N;
69         if (k % N == N - 1)
70             z[2] = k - (h * N + N - 1);
71         else
72             z[2] = k - h * N + 1;
73         z[1] = k - (h - 1) * N;
74     } else {
75         if (k % N == N - 1) {
76             z[0] = k - (h + 2) * N + 1;
77             z[2] = k - (h + 1) * N + 1;
78         } else {
79             z[0] = k - (h + 1) * N + 1;
80             z[2] = k - h * N + 1;
81         }
82     }
83 }

```

```

82         z[1] = k - h * N;
83     }
84 }
85
86 void number_u(int i, int z[6])
87 {
88     int j, r, s;
89
90     s = i % N2;
91
92     if (s < N) {
93         r = i / N2;
94         j = N4 * r + 2 * s;
95
96         z[0] = j;
97         z[1] = j + N4;
98         if (s == N - 1)
99             z[2] = j + 2 - N2;
100        else
101            z[2] = j + 2;
102        z[3] = j + N2 + 1;
103        z[4] = j + 1;
104        z[5] = z[3] - 1;
105    } else {
106        r = i / N2;
107        s = s - N;
108        j = N4 * r + 2 * s;
109
110        z[1] = j + N4;
111        z[3] = z[1] + 1;
112        z[5] = j + N2 + 1;
113
114        if (s == N - 1) {
115            z[0] = j + 2 - N2;
116            z[2] = j + N2 + 2;
117            z[4] = j + 2;
118        } else {
119            z[0] = j + 2;
120            z[2] = z[1] + 2;
121            z[4] = z[5] + 1;
122        }
123    }
124 }
125
126 double cal_r(int n, double r)
127 {
128     if (n == 1) {
129         return (r - R1) * (r - R1) / (R2 - R1) + R1;
130     } else if (n == 2) {
131         return sqrt((r - R1) * (R2 - R1)) + R1;
132     } else
133         return r;
134 }
135
136 double cal_x(int k)
137 {
138     int i, j;
139     double Pi, r, theta, dx;

```

```

140
141     dx = 0.0;
142     Pi = 4.0 * atan(1.0);
143
144     i = k / N;
145     j = k % N;
146
147     r = (R2 - R1) * i / M1 + R1;
148     theta = 2 * Pi * j / N;
149
150     /* return cal_r(C,r)*cos(theta) + (R2 - r)/(R2 - R1)*dx;*/
151     return cal_r(C, r) * cos(theta) + (R2 - cal_r(C, r)) / (R2 - R1) * dx;
152 }
153
154 double cal_y(int k)
155 {
156     int i, j;
157     double Pi, r, theta;
158
159     Pi = 4.0 * atan(1.0);
160
161     i = k / N;
162     j = k % N;
163
164     r = (R2 - R1) * i / M1 + R1;
165     theta = 2 * Pi * j / N;
166
167     return cal_r(C, r) * sin(theta);
168 }
169
170 double blu(int func, double theta, double R1,
171           double mu, double omega1, double A , double B)
172 {
173     double Pi, b1, ur, ut;
174
175     Pi = 4.0 * atan(1.0);
176     b1 = 0.0;
177
178     if (func == 0) {
179         ur = mu / R1;
180         ut = b1;
181     }
182     else if (func == 1) {
183         if (theta <= Pi / 2.0)
184             ur = 2 * sin(2 * theta) / R1;
185         else
186             ur = 0.0;
187
188         ut = 0.0;
189     }
190     else if (func == 2) {
191         if (theta <= Pi / 2.0)
192             ur = 4.0 / R1;
193         else if (Pi <= theta && theta <= 1.5 * Pi)
194             ur = -2.0 / R1;
195         else
196             ur = 0.0;
197

```

```

198         ut = 0.0;
199     }
200
201     else if(func == 3){
202         ur = mu / R1 + A * R1 * cos(theta);
203         ut = R1 * omega1 + B * R1 * sin(theta);
204     }
205
206     else {
207         fprintf(stderr, "b1u: func is out of range.\n");
208         exit(-1);
209     }
210     return ur * cos(theta) - ut * sin(theta);
211 }
212
213 double b2u(int func, double theta, double R2,
214           double mu, double omega2, double A, double B)
215 {
216     double Pi, b2, ur, ut;
217
218     Pi = 4.0 * atan(1.0);
219     b2 = 1.0;
220
221     if (func == 0) {
222         ur = mu / R2;
223         ut = b2;
224     }
225     else if (func == 1) {
226         if ((theta >= Pi) && (theta <= 1.5 * Pi))
227             ur = 2 * sin(2 * theta) / R2;
228         else
229             ur = 0.0;
230
231         ut = 0.0;
232     }
233     else if (func == 2) {
234         if ((theta >= 0.5 * Pi) && (theta <= Pi))
235             ur = 2.0 / R2;
236         else
237             ur = 0.0;
238
239         ut = 0.0;
240     }
241
242     else if(func == 3){
243         ur = mu / R2 + A * R2 * cos(theta);
244         ut = R2 * omega2 + B * R2 * sin(theta);
245     }
246
247     else {
248         fprintf(stderr, "b2u: func is out of range.\n");
249         exit(-1);
250     }
251     return ur * cos(theta) - ut * sin(theta);
252 }
253
254 double b1v(int func, double theta, double R1,
255           double mu, double omega1, double A, double B)

```

```

256 {
257     double Pi, b1, ur, ut;
258
259     Pi = 4.0 * atan(1.0);
260     b1 = 0.0;
261
262     if (func == 0) {
263         ur = mu / R1;
264         ut = b1;
265     }
266     else if (func == 1) {
267         if (theta <= Pi / 2.0)
268             ur = 2 * sin(2 * theta) / R1;
269         else
270             ur = 0.0;
271
272         ut = 0.0;
273     }
274     else if (func == 2) {
275         if (theta <= Pi / 2.0)
276             ur = 4.0 / R1;
277         else if (Pi <= theta && theta <= 1.5 * Pi)
278             ur = -2.0 / R1;
279         else
280             ur = 0.0;
281
282         ut = 0.0;
283     }
284
285     else if(func == 3){
286         ur = mu / R1 + A * R1 * cos(theta);
287         ut = R1 * omega1 + B * R1 * sin(theta);
288     }
289
290     else {
291         fprintf(stderr, "b1v: func is out of range.\n");
292         exit(-1);
293     }
294     return ur * sin(theta) + ut * cos(theta);
295 }
296
297 double b2v(int func, double theta, double R2,
298           double mu, double omega2, double A, double B)
299 {
300     double Pi, b2, ur, ut;
301
302     Pi = 4.0 * atan(1.0);
303     b2 = 1.0;
304
305     if (func == 0) {
306         ur = mu / R2;
307         ut = b2;
308     }
309     else if (func == 1) {
310         if ((theta >= Pi) && (theta <= 1.5 * Pi))
311             ur = 2 * sin(2 * theta) / R2;
312         else
313             ur = 0.0;

```

```

314         ut = 0.0;
315     }
316     else if (func == 2) {
317         if ((theta >= 0.5 * Pi) && (theta <= Pi))
318             ur = 2.0 / R2;
319         else
320             ur = 0.0;
321     }
322     ut = 0.0;
323 }
324
325 else if(func == 3){
326     ur = mu / R2 + A * R2 * cos(theta);
327     ut = R2 * omega2 + B * R2 * sin(theta);
328 }
329
330 else {
331     fprintf(stderr, "b2v: func is out of range.\n");
332     exit(-1);
333 }
334 return ur * sin(theta) + ut * cos(theta);
335 }
336
337 void inver1(double *u, double *b, int j)
338 {
339     int i, p, q;
340
341     for (i = 0; i < N; i++) {
342         p = 5 * i + N9 * j;
343         q = 2 * i + N4 * j;
344         u[q] = b[p];
345         u[q + U] = b[p + 1];
346         u[i + N * j + U2] = b[p + 2];
347         u[q + 1] = b[p + 3];
348         u[q + 1 + U] = b[p + 4];
349     }
350 }
351
352 void inver2(double *u, double *b, int j)
353 {
354     int i, p, q;
355
356     for (i = 0; i < N2; i++) {
357         p = 2 * i + N9 * j + P;
358         q = i + N4 * j + N2;
359         u[q] = b[p];
360         u[q + U] = b[p + 1];
361     }
362 }
363
364 int main(int argc, char **argv)
365 {
366     FILE *f;
367     int i, j, k, zp[3], zu[6], func, forc;
368     double nu, rho, Pi, h, theta;
369     double *x, *y;
370     double mu, omega1, omega2, A, B;

```



```

372
373     init();
374     x = malloc(sizeof(double) * U);
375     y = malloc(sizeof(double) * U);
376     if (argc != 2) {
377         printf("usage: %s <output file>\n", argv[0]);
378         exit(1);
379     }
380     if ((f = fopen(argv[1], "w")) == NULL) {
381         fprintf(stderr, "can't open %s \n", argv[1]);
382         exit(1);
383     }
384
385     printf("ν, ρ =");
386     scanf("%lf %lf", &nu, &rho);
387     printf("func forc = ");
388     scanf("%d %d", &func, &forc);
389
390     if(func == 3){
391         printf("境界条件 b において mu ,omega1, omega2 ,A ,B の入力を行う\n");
392         printf("mu = ");
393         scanf("%lf", &mu);
394         printf("角速度 omega1, omega2 = ");
395         scanf("%lf %lf", &omega1, &omega2);
396         printf("摂動係数 A, B = ");
397         scanf("%lf %lf", &A, &B);
398     }
399
400     fprintf(f, "%4d   %4d \n", T, U);
401     fprintf(f, "\n");
402
403     for (i = 0; i < T; i++) {
404         number_p(i, zp);
405         number_u(i, zu);
406         for (j = 0; j < 3; j++) {
407             x[zu[j]] = cal_x(zp[j]);
408             y[zu[j]] = cal_y(zp[j]);
409         }
410         x[zu[3]] = 0.5 * (x[zu[1]] + x[zu[2]]);
411         y[zu[3]] = 0.5 * (y[zu[1]] + y[zu[2]]);
412         x[zu[4]] = 0.5 * (x[zu[0]] + x[zu[2]]);
413         y[zu[4]] = 0.5 * (y[zu[0]] + y[zu[2]]);
414         x[zu[5]] = 0.5 * (x[zu[0]] + x[zu[1]]);
415         y[zu[5]] = 0.5 * (y[zu[0]] + y[zu[1]]);
416     }
417
418     for (i = 0; i < U; i++) {
419         fprintf(f, "%25.20f  %25.20f \n", x[i], y[i]);
420     }
421     fprintf(f, "\n");
422
423     for (i = 0; i < T; i++) {
424         number_u(i, zu);
425         for (j = 0; j < 6; j++) {
426             fprintf(f, " %4d ", zu[j]);
427         }
428         fprintf(f, "\n");
429     }

```

```

430     fprintf(f, "\n");
431
432     k = 0;
433     for (i = 0; i < U; i++) {
434         fprintf(f, "%4d \n", k);
435         if (i % N4 < N2 && (i % 2) == 0)
436             j = 3;
437         else
438             j = 2;
439         k += j;
440     }
441     fprintf(f, "\n");
442
443     fprintf(f, "%4d \n", k);
444     fprintf(f, "\n");
445
446     fprintf(f, "%f %f \n", nu, rho);
447     fprintf(f, "\n");
448
449     fprintf(f, " %4d 0 \n", N4);
450     fprintf(f, "\n");
451
452     Pi = 4.0 * atan(1.0);
453     h = Pi / N;
454
455     for (i = 0; i < N2; i++) {
456         theta = i * h;
457         j = U - N2 + i;
458         fprintf(f, "%4d %25.20f %25.20f\n", i,
459             b1u(func, theta, R1, mu, omega1, A, B),
460             b1v(func, theta, R1, mu, omega1, A, B));
461         fprintf(f, "%4d %25.20f %25.20f\n", j,
462             b2u(func, theta, R2, mu, omega2, A, B),
463             b2v(func, theta, R2, mu, omega2, A, B));
464     }
465     fprintf(f, "\n");
466
467     if (forc == 0) {
468         for (i = 0; i < U; i++)
469             fprintf(f, "%25.20f %25.20f\n", 0.0, 0.0);
470     }
471     else {
472         for (i = 0; i < U; i++) {
473             fprintf(f, "%25.20f %25.20f\n", x[i], y[i]);
474         }
475     }
476     fclose(f);
477     return 0;
478 }
479
480

```

## 6.2 Stokes 方程式を解くプログラム

1 /\*

```

2  * stokes-fukushima.c
3  *
4  *   工藤氏作成のプログラム（1995年2月）を大幅
5  *   第1段
6  *   0. ANSI C のプロトタイプ宣言をつけた
7  *   1. 注釈を加えた
8  *   2. 変数の初期化忘れを直した
9  *   第2段
10 *   LAPACK の dgbstv を使うようにした。
11 *   第3段
12 *   手計算で詳しく求めた要素係数行列の公式を利用するように書き改めた
13 *   （福島）
14 */
15
16 #include <math.h>
17 #include <stdio.h>
18 #include "matutil.h"
19
20 int verbose = 0;
21
22 /* プロトタイプ宣言 */
23 double menseki(double, double, double, double, double, double);
24 void make(int,
25           vector, int, int,
26           vector, double, double,
27           vector, vector, int **, int *,
28           vector, vector);
29 void change(int, vector, int, int, int);
30 void bound(vector, int, int,
31           vector, int *, vector, vector, int, int, int *, int);
32 void etalu(matrix, vector, int);
33
34 void dgbstv_(int *n, int *kl, int *ku, int *nrhs,
35             double *AB, int *ldab, int *ipiv, double *b, int *ldb, int *info);
36 int max(int a, int b) { return (a > b) ? a : b; }
37 int min(int a, int b) { return (a < b) ? a : b; }
38 /* 一次元配列で計算するのは生で書くと大変なのでマクロで処理 */
39 #define A(i,j) ab[(j)*(nb)+(2*(nband)+(i)-(j))]
40
41 int main(int argc, char **argv)
42 {
43     /****** 変数宣言 *****/
44     int i, j;
45     /* 入力ファイル, 出力ファイル */
46     FILE *f1, *f2;
47     /* nnode: 節点の総数, nelmt: 要素の総数, SIZE: 未知数の総数 */
48     int nnode, nelmt, SIZE;
49     /* nNBC: Neumann B.C. を課す節点の総数,
50      * nDBC: Dirichlet B.C. を課す節点の総数
51      * ND == nDBC + nNBC */
52     int nNBC, nDBC, ND;
53     /*
54      * cum[nnode]: 累積節点番号表,
55      * num[nelmt][6]: 要素節点番号表 (各要素を構成する節点の全体節点番号の表)
56      * b[]: 境界データ (Dirichlet と Neumann)
57      */
58     int *cum, **num, *b;
59     /* nu: 粘性係数  $\nu$ , rho: 密度  $\rho$  */

```

```

60     double nu, rho;
61     /* x[nnode], y[nnode]: 節点の座標,
62        bx[ND], by[ND]: 境界データ,
63        fx[], fy[]: 外力データ,
64        sx[], sy[]: 応力テンソル,
65        ab[]: 全体剛性行列 */
66     vector x, y, u, bx, by, fx, fy, sx, sy, ab;
67     ivector ipvt;
68     /* 半バンド幅 */
69     int nband, nb;
70     /* */
71     char *input_file, *output_file;
72     /***** 変数宣言終了, 以下実行文 *****/
73
74     /* 引数の個数をチェック */
75     if (argc == 3) {
76         input_file = argv[1];
77         output_file = argv[2];
78     }
79     else {
80 #ifdef ORIGINAL
81         fprintf(stderr, "usage: %s <input_file_name> <output_file_name>\n",
82                argv[0]);
83         exit(1);
84 #else
85         input_file = "reidai.data";
86         output_file = "fukushima.out";
87 #endif
88     }
89     /* 入力ファイルを開く */
90     if ((f1 = fopen(input_file, "r")) == NULL) {
91         fprintf(stderr, "Can't open %s\n", input_file);
92         exit(1);
93     }
94     /* 出力ファイルを開く (オリジナルでは計算後にあったが、前に開くべきだ) */
95     if ((f2 = fopen(output_file, "w")) == NULL) {
96         fprintf(stderr, "Can't open %s\n", output_file);
97         exit(2);
98     }
99     /* 要素数 nelmt, 節点数 nnode を読み込む */
100    fscanf(f1, "%d %d", &nelmt, &nnode);
101    if (verbose)
102        printf("要素数=%d, 総節点数=%d\n", nelmt, nnode);
103
104    /* 節点の座標 (x,y) を読み込む */
105    if (verbose)
106        printf("節点の座標を読む。 \n");
107    x = new_vector(nnode); y = new_vector(nnode);
108    if (x == NULL || y == NULL) {
109        fprintf(stderr, "節点の座標用のメモリーが足りません。 \n");
110        exit(1);
111    }
112    for (i = 0; i < nnode; i++)
113        fscanf(f1, "%lf %lf", &x[i], &y[i]);
114
115    if (verbose)
116        printf("要素節点番号表を読む。 \n");
117    /* 要素節点番号表 num[nelmt][6] を確保 */

```

```

118     num = malloc(nelmt * sizeof(int *));
119     if (num == NULL) {
120         fprintf(stderr, "要素と節点の対応表用のメモリーが足りません\n");
121         exit(1);
122     }
123     for (i = 0; i < nelmt; i++) {
124         num[i] = malloc(sizeof(int) * 6);
125         if (num[i] == NULL) {
126             fprintf(stderr, "要素と節点の対応用のメモリーが足りません\n");
127             exit(1);
128         }
129     }
130     /* num[nelmt][6] を読み込む */
131     for (i = 0; i < nelmt; i++)
132         for (j = 0; j < 6; j++)
133             fscanf(f1, "%d", &num[i][j]);
134
135     if (verbose)
136         printf("累積節点番号表を読む。 \n");
137     /* 累積節点番号表 cum[nnode] を確保 */
138     if ((cum = (int *) malloc(sizeof(int) * nnode)) == NULL) {
139         fprintf(stderr, "累積節点番号用のメモリーが足りません \n");
140         exit(1);
141     }
142     /* cum[nnode] を読み込む */
143     for (i = 0; i < nnode; i++)
144         fscanf(f1, "%d", &cum[i]);
145
146     /* 未知数の総数 SIZE を読み込む */
147     fscanf(f1, "%d", &SIZE);
148     if (verbose)
149         printf("未知数の個数=%d\n", SIZE);
150
151     /* 半バンド幅 nband の評価 */
152     {
153         int max_cum, min_cum, cum_tmp, band_tmp;
154         if (verbose)
155             printf("半バンド幅の評価\n");
156         nband = 0;
157         for (i = 0; i < nelmt; i++) {
158             max_cum = 0;
159             min_cum = SIZE;
160             for (j = 0; j < 6; j++) {
161                 cum_tmp = cum[num[i][j]];
162                 if (max_cum < cum_tmp) max_cum = cum_tmp;
163                 if (min_cum > cum_tmp) min_cum = cum_tmp;
164             }
165             band_tmp = max_cum - min_cum;
166             if (band_tmp > nband)
167                 nband = band_tmp;
168         }
169         if (verbose)
170             printf("半バンド幅=%d\n", nband);
171     }
172
173     /* 安心 */
174     nband += 2;
175     nb = 3 * nband + 1;

```

```

176
177 /* 連立1次方程式  $Ax = u$  のための変数  $ab[nb*SIZE]$ ,  $u[SIZE]$  を確保 */
178 u = new_vector(SIZE);
179 ab = new_vector(nb * SIZE);
180 ipvt = new_ivector(SIZE);
181 if (u == NULL || ab == NULL || ipvt == NULL) {
182     fprintf(stderr, "近似解用のメモリーが足りません。 \n");
183     exit(1);
184 }
185 for (i = 0; i < SIZE; i++)
186     u[i] = 0.0;
187 for (i = 0; i < (nb * SIZE); i++)
188     ab[i] = 0.0;
189
190 /* 粘性係数 nu, 密度 rho を読み込む */
191 fscanf(f1, "%lf %lf", &nu, &rho);
192 /* Dirichlet 境界, Neumann 境界上の節点の総数 nNBC, nDBC を読み込む */
193 fscanf(f1, "%d %d", &nNBC, &nDBC);
194
195 /* 境界上の接点の総数 ND を計算 */
196 ND = nNBC + nDBC;
197
198 /* 境界データを記憶する bx[ND], by[ND] を用意し、読み込む */
199 if ((b = malloc(sizeof(int) * ND)) == NULL) {
200     fprintf(stderr, "境界に属する節点番号表用のメモリーが足りません。 \n");
201     exit(1);
202 }
203 bx = new_vector(ND); by = new_vector(ND);
204 if (bx == NULL || by == NULL) {
205     fprintf(stderr, "境界条件データ用のメモリーが足りません。 \n");
206     exit(1);
207 }
208 for (i = 0; i < ND; i++)
209     fscanf(f1, "%d %lf %lf", &b[i], &bx[i], &by[i]);
210
211 /* Neumann 境界データを表わす sx[nnode], sy[nnode] を用意 (豪勢な使い方だ) */
212 sx = new_vector(nnode); sy = new_vector(nnode);
213 if (sx == NULL || sy == NULL) {
214     fprintf(stderr, "メモリーが足りません。 \n");
215     exit(1);
216 }
217 for (i = 0; i < nnode; i++)
218     sx[i] = sy[i] = 0.0;
219 if (nNBC != 0) { /* ... Neumann 境界が空でないならば */
220     for (i = nDBC; i < ND; i++) { /* 注: 最初の nDBC 個は Dirichlet データ */
221         sx[b[i]] = bx[i];
222         sy[b[i]] = by[i];
223     }
224 }
225 /* 外力 fx[nnode], fy[nnode] を確保し、読み込む */
226 fx = new_vector(nnode); fy = new_vector(nnode);
227 if (fx == NULL || fy == NULL) {
228     fprintf(stderr, "外力データ用のメモリーが足りません。 \n");
229     exit(1);
230 }
231 for (i = 0; i < nnode; i++)
232     fscanf(f1, "%lf %lf", &fx[i], &fy[i]);
233

```

```

234     /* 入力データ読み込み終了 */
235     fclose(f1);
236
237     /* 連立 1 次方程式を作る */
238     make(nelmt, ab, nband, nb, u, nu, rho, x, y, num, cum, fx, fy);
239     bound(ab, nband, nb, u, b, bx, by, nNBC, nDBC, cum, SIZE);
240
241     /* 不要になったメモリーを返却 */
242     free(num);
243     free_vector(x);
244     free_vector(y);
245     free_vector(fx);
246     free_vector(fy);
247
248     free(b);
249     free(cum);
250     free_vector(bx);
251     free_vector(by);
252
253     /* 連立 1 次方程式を解く */
254     {
255         int nrhs = 1, ldab = nb, ldb = SIZE, info;
256         dgbsv_(&SIZE, &nband, &nband, &nrhs, ab, &ldab, ipvt, u, &ldb, &info);
257         if (info == 0) {
258             if (verbose)
259                 printf(" successful\n");
260         }
261         else if (info > 0)
262             printf(" U is singular\n");
263         else
264             printf("%d-th argument has illegal value.\n", abs(info));
265     }
266
267     /* 計算結果をファイル f2 に出力し、ファイルを閉じる */
268     for (i = 0; i < SIZE; i++)
269         fprintf(f2, "%f \n", u[i]);
270     fclose(f2);
271
272     return 0;
273 }
274
275 /* 三角形の頂点の座標から (符号付きの) 面積を計算する */
276 double menseki(double x0, double x1, double x2,
277                double y0, double y1, double y2)
278 {
279     double s;
280     s = 0.5 * ((x1 - x0) * (y2 - y0) - (y1 - y0) * (x2 - x0));
281     return s;
282 }
283
284 /*弱形式を表わす連立 1 次方程式を作る。
285  * 菊地流プログラムで言うと ecm(), assem() を合わせた内容
286  */
287 void make(int T,
288           vector ab, int nband, int nb,
289           vector u, double nu, double rho,
290           vector x, vector y, int **num, int *cum,
291           vector fx, vector fy)

```

```

292 {
293     int i, j, k;
294     double a0, a1, a2, b0, b1, b2, c0, c1, c2, S, D, nurho;
295
296     matrix a = new_matrix(6, 6);
297     matrix b = new_matrix(6, 3);
298     matrix c = new_matrix(6, 3);
299     matrix d = new_matrix(6, 6);
300
301     nurho = nu * rho;
302
303     for (k = 0; k < T; k++) { /* 各要素について */
304         int n0, n1, n2;
305         n0 = num[k][0];
306         n1 = num[k][1];
307         n2 = num[k][2];
308         S = menseki(x[n0], x[n1], x[n2], y[n0], y[n1], y[n2]);
309         if (S < 0.0) {
310             printf("第 %d 要素の面積が負になっています!\n", k);
311             exit(0);
312         }
313         D = 2 * S;
314
315         a0 = (y[n1] - y[n2]) / D;
316         a1 = (y[n2] - y[n0]) / D;
317         a2 = (y[n0] - y[n1]) / D;
318         b0 = (x[n2] - x[n1]) / D;
319         b1 = (x[n0] - x[n2]) / D;
320         b2 = (x[n1] - x[n0]) / D;
321
322         c0 = a1 * a2 + b1 * b2;
323         c1 = a2 * a0 + b2 * b0;
324         c2 = a0 * a1 + b0 * b1;
325
326         a[0][0] = 3 * (a0 * a0 + b0 * b0);
327         a[1][1] = 3 * (a1 * a1 + b1 * b1);
328         a[2][2] = 3 * (a2 * a2 + b2 * b2);
329         a[0][1] = -c2;
330         a[0][2] = -c1;
331         a[1][2] = -c0;
332         a[0][3] = a[1][4] = a[2][5] = 0;
333         a[1][3] = a[2][3] = 4 * c0;
334         a[0][4] = a[2][4] = 4 * c1;
335         a[0][5] = a[1][5] = 4 * c2;
336         a[3][3] = a[4][4] = a[5][5] = -8 * (c0 + c1 + c2);
337         a[3][4] = 8 * c2;
338         a[3][5] = 8 * c1;
339         a[4][5] = 8 * c0;
340
341         for(i = 0; i < 6; i++)
342             for(j = 0; j <= i; j++)
343                 a[i][j] = a[j][i];
344
345         for(i = 0; i < 6; i++)
346             for(j = 0; j < 6; j++)
347                 a[i][j] = (S / 3.0) * a[i][j];
348
349         b[0][0] = a0;

```



```

350     b[1][1] = a1;
351     b[2][2] = a2;
352     b[0][1] = b[0][2] = b[1][0] = b[1][2] = b[2][0] = b[2][1] = 0;
353     b[3][0] = a1 + a2;
354     b[4][1] = a2 + a0;
355     b[5][2] = a0 + a1;
356     b[3][1] = a1 + 2 * a2;
357     b[3][2] = 2 * a1 + a2;
358     b[4][0] = 2 * a2 + a0;
359     b[4][2] = a2 + 2 * a0;
360     b[5][0] = a0 + 2 * a1;
361     b[5][1] = 2 * a0 + a1;
362
363     for(i = 0; i < 6; i++)
364         for(j = 0; j < 3; j++)
365             b[i][j] = (S / 3.0) * b[i][j];
366
367     c[0][0] = b0;
368     c[1][1] = b1;
369     c[2][2] = b2;
370     c[0][1] = c[0][2] = c[1][0] = c[1][2] = c[2][0] = c[2][1] = 0;
371     c[3][0] = b1 + b2;
372     c[4][1] = b2 + b0;
373     c[5][2] = b0 + b1;
374     c[3][1] = b1 + 2 * b2;
375     c[3][2] = 2 * b1 + b2;
376     c[4][0] = 2 * b2 + b0;
377     c[4][2] = b2 + 2 * b0;
378     c[5][0] = b0 + 2 * b1;
379     c[5][1] = 2 * b0 + b1;
380
381     for(i = 0; i < 6; i++)
382         for(j = 0; j < 3; j++)
383             c[i][j] = (S / 3.0) * c[i][j];
384
385     d[0][0] = d[1][1] = d[2][2] = 6;
386     d[0][1] = d[0][2] = d[1][2] = -1;
387     d[0][3] = d[1][4] = d[2][5] = -4;
388     d[0][4] = d[0][5] = d[1][3] = d[1][5] = d[2][3] = d[2][4] = 0;
389     d[3][3] = d[4][4] = d[5][5] = 32;
390     d[3][4] = d[3][5] = d[4][5] = 16;
391
392     for(i = 0; i < 6; i++)
393         for(j = 0; j <= i; j++)
394             d[i][j] = d[j][i];
395
396     for(i = 0; i < 6; i++)
397         for(j = 0; j < 6; j++)
398             d[i][j] = (S / 180.0) * d[i][j];
399
400     /* 直接刚性法 */
401     for(i = 0; i < 6; i++) {
402         int I = cum[num[k][i]];
403         for (j = 0; j < 6; j++) {
404             int J = cum[num[k][j]];
405             A(I,J) += nurho * a[i][j];
406             A(I+1,J+1) += nurho * a[i][j];
407         }

```

```

408
409         for (j = 0; j < 3; j++) {
410             int J = cum[num[k][j]];
411             A(I,J+2) -= b[i][j];
412             A(I+1,J+2) -= c[i][j];
413         }
414
415         for(j = 0; j < 6; j++){
416             u[I] += rho * d[i][j] * fx[num[k][j]];
417             u[I + 1] += rho * d[i][j] * fy[num[k][j]];
418         }
419     }
420
421     for(i = 0; i < 3; i++) {
422         int I = cum[num[k][i]];
423         for (j = 0; j < 6; j++) {
424             int J = cum[num[k][j]];
425             A(I+2,J) -= b[j][i];
426             A(I+2,J+1) -= c[j][i];
427         }
428     }
429 }
430
431 /* 不要なメモリーを返却 */
432 free_matrix(a);
433 free_matrix(b);
434 free_matrix(c);
435 free_matrix(d);
436 }
437
438 void change(int n, vector ab, int nband, int nb, int k)
439 {
440     int i,j;
441     for (i = max(0,k-nband); i <= min(n-1,k+nband); i++)
442         A(i,k) = 0.0;
443     for (j = max(0,k-nband); j <= min(n-1,k+nband); j++)
444         A(k,j) = 0.0;
445     A(k,k) = 1.0;
446 }
447
448 void bound(vector ab, int nband, int nb,
449           vector u, int *b, vector bx, vector by,
450           int N, int D, int *cum, int n)
451 {
452     int i, j, k;
453     for (k = 0; k < D; k++) {
454         j = cum[b[k]];
455         for (i = max(0,j-nband); i <= min(n-1,j+nband); i++)
456             u[i] -= A(i,j) * bx[k];
457         u[j] = bx[k];
458         change(n, ab, nband, nb, j);
459         j++;
460         for (i = max(0,j-nband); i <= min(n-1,j+nband); i++)
461             u[i] -= A(i,j) * by[k];
462         u[j] = by[k];
463         change(n, ab, nband, nb, j);
464     }
465     if (N == 0) {

```

```

466     change(n, ab, nband, nb, 2);
467     u[2] = 0.0;
468 }
469 }
470
471
472
473
474
475

```

### 6.3 行列 $\kappa_{ijk}^x, \kappa_{ijk}^y$ の計算プログラム (Mathematicaによる)

```

1  lambda[0] := 1 - x - y
2  lambda[1] := x
3  lambda[2] := y
4  phi[0] := lambda[0](2lambda[0] - 1)
5  phi[1] := lambda[1](2lambda[1] - 1)
6  phi[2] := lambda[2](2lambda[2] - 1)
7  phi[3] := 4lambda[1]lambda[2]
8  phi[4] := 4lambda[0]lambda[2]
9  phi[5] := 4lambda[0]lambda[1]
10 phix[0] := (4lambda[0] - 1)alpha[0]
11 phix[1] := (4lambda[1] - 1)alpha[1]
12 phix[2] := (4lambda[2] - 1)alpha[2]
13 phix[3] := 4(alpha[1]lambda[2] + alpha[2]lambda[1])
14 phix[4] := 4(alpha[2]lambda[0] + alpha[0]lambda[2])
15 phix[5] := 4(alpha[0]lambda[1] + alpha[1]lambda[0])
16 kappa[i, j, k] := 2*Integrate[Integrate[phi[i]phi[j]phix[k],
17     {y, 0, 1 - x}], {x, 0, 1}]
18 kappa[0] := Table[kappa[i, j, 0], {i, 0, 5}, {j, 0, 5}]
19 kappa[1] := Table[kappa[i, j, 1], {i, 0, 5}, {j, 0, 5}]
20 kappa[2] := Table[kappa[i, j, 2], {i, 0, 5}, {j, 0, 5}]
21 kappa[3] := Table[kappa[i, j, 3], {i, 0, 5}, {j, 0, 5}]
22 kappa[4] := Table[kappa[i, j, 4], {i, 0, 5}, {j, 0, 5}]
23 kappa[5] := Table[kappa[i, j, 5], {i, 0, 5}, {j, 0, 5}]

```

### 6.4 Navier-Stokes 方程式を解くプログラム

```

1  /*
2  *  navier3-fukushima.c
3  *   工藤氏作成のプログラム (1995年2月) を若干修正
4  *   0. ANSI C のプロトタイプ宣言をつけた
5  *   1. 注釈を加えた (ここまで桂田ス・砲茲)
6  *   2. 新たに求めた要素係数行列に対応させた.
7  */
8
9  #include <math.h>
10 #include <stdio.h>
11 #include "matutil.h"
12
13 int verbose = 0;

```

```

14
15 double menseki(double x0, double x1, double x2,
16               double y0, double y1, double y2);
17 void matprod3(matrix a, matrix b, matrix c, matrix d);
18 void make(int T, vector ab, vector ab0, int nband, int nb, int SIZE,
19          vector u, vector v, double nu, double rho,
20          vector x, vector y, int **num, int *cum,
21          vector fx, vector fy);
22 void change(int n, vector ab, int nband, int nb, int k);
23 void bound(vector ab, int nband, int nb,
24           vector u, int *b, int N, int D, int *cum, int SIZE);
25 void newton(int T,
26            int nband, int nb, ivector ipvt,
27            vector u, vector v, double nu, double rho,
28            vector x, vector y, int **num, int *cum,
29            vector fx, vector fy, int *b, int N, int D, int SIZE);
30
31 void dgbsv_(int *n, int *kl, int *ku, int *nrhs,
32            double *AB, int *ldab, int *ipiv, double *b, int *ldb, int *info);
33 int max(int a, int b) { return (a > b) ? a : b; }
34 int min(int a, int b) { return (a < b) ? a : b; }
35 /* 一次元配列で計算するのは生で書くと大変なのでマクロで処理 */
36 #define A0(i,j) ab0[(j)*(nb)+(2*(nband)+(i)-(j))]
37 #define A(i,j) ab[(j)*(nb)+(2*(nband)+(i)-(j))]
38
39 int main(int argc, char **argv)
40 {
41     FILE *f1, *f2, *f3;
42     int i, j, nnode, nelmt, SIZE, N, D, ND;
43     int *cum, **num, *b;
44     double nu, rho;
45     vector x, y, u, v, bx, by, fx, fy, sx, sy;
46     ivector ipvt;
47     int nband, nb;
48
49     if (argc != 4) {
50         fprintf(stderr, "usage: %s <input file0> <input file1> <output file>",
51                argv[0]);
52         exit(1);
53     }
54     if ((f1 = fopen(argv[1], "r")) == NULL) {
55         fprintf(stderr, "Can't open %s", argv[1]);
56         exit(1);
57     }
58     if ((f2 = fopen(argv[2], "r")) == NULL) {
59         fprintf(stderr, "Can't open %s", argv[2]);
60         exit(2);
61     }
62
63     fscanf(f1, "%d %d", &nelmt, &nnode);
64
65     x = new_vector(nnode);
66     y = new_vector(nnode);
67     for (i = 0; i < nnode; i++) {
68         fscanf(f1, "%lf %lf", &x[i], &y[i]);
69     }
70
71     num = malloc(nelmt * sizeof(void *));

```

```

72  if (num == NULL) {
73      fprintf(stderr, "要素と節点の対応用のメモリーが足りません\n");
74      exit(1);
75  }
76  for (i = 0; i < nelmt; i++) {
77      num[i] = malloc(sizeof(int) * 6);
78      if (num[i] == NULL) {
79          fprintf(stderr, "要素と節点の対応用のメモリーが足りません\n");
80          exit(1);
81      }
82  }
83  for (i = 0; i < nelmt; i++)
84      for (j = 0; j < 6; j++)
85          fscanf(f1, "%d", &num[i][j]);
86
87  if ((cum = malloc(sizeof(int) * nnode)) == NULL) {
88      fprintf(stderr, "累積節点番号用のメモリーが足りません\n");
89      exit(1);
90  }
91  for (i = 0; i < nnode; i++)
92      fscanf(f1, "%d", &cum[i]);
93  fscanf(f1, "%d", &SIZE);
94  if (verbose)
95      printf("未知数の個数=%d\n", SIZE);
96
97  /* 半バンド幅 nband の評価 */
98  {
99      int max_cum, min_cum, cum_tmp, band_tmp;
100     if (verbose)
101         printf("半バンド幅の評価\n");
102     nband = 0;
103     for (i = 0; i < nelmt; i++) {
104         max_cum = 0;
105         min_cum = SIZE;
106         for (j = 0; j < 6; j++) {
107             cum_tmp = cum[num[i][j]];
108             if (max_cum < cum_tmp) max_cum = cum_tmp;
109             if (min_cum > cum_tmp) min_cum = cum_tmp;
110         }
111         band_tmp = max_cum - min_cum;
112         if (band_tmp > nband)
113             nband = band_tmp;
114     }
115     if (verbose)
116         printf("半バンド幅=%d\n", nband);
117 }
118 /* 安心 */
119 nband += 2;
120 nb = 3 * nband + 1;
121
122 u = new_vector(SIZE);
123 v = new_vector(SIZE);
124 ipvt = new_ivector(SIZE);
125
126 for (i = 0; i < SIZE; i++)
127     fscanf(f2, "%lf", &u[i]);
128 fclose(f2);
129

```

```

130  fscanf(f1, "%lf %lf", &nu, &rho);
131  fscanf(f1, "%d %d", &D, &N);
132
133  ND = N + D;
134
135  if ((b = malloc(sizeof(int) * ND)) == NULL) {
136      fprintf(stderr, "境界の番号用のメモリーが足りません。 \n");
137      exit(1);
138  }
139  bx = new_vector(ND);
140  by = new_vector(ND);
141  for (i = 0; i < ND; i++)
142      fscanf(f1, "%d %lf %lf", &b[i], &bx[i], &by[i]);
143
144  sx = new_vector(nnode);
145  sy = new_vector(nnode);
146  for (i = 0; i < nnode; i++)
147      sx[i] = sy[i] = 0.0;
148  if (N != 0) {
149      for (i = D; i < ND; i++) {
150          sx[b[i]] = bx[i];
151          sy[b[i]] = by[i];
152      }
153  }
154
155  fx = new_vector(nnode);
156  fy = new_vector(nnode);
157  for (i = 0; i < nnode; i++)
158      fscanf(f1, "%lf %lf", &fx[i], &fy[i]);
159  fclose(f1);
160
161  newton(nelmt, nband, nb, ipvt,
162        u, v, nu, rho, x, y, num, cum, fx, fy, b, N, D,
163        SIZE);
164
165  free(num);
166  free_vector(x);
167  free_vector(y);
168  free_vector(fx);
169  free_vector(fy);
170
171  free(b);
172  free(cum);
173  free_vector(bx);
174  free_vector(by);
175  free_vector(sx);
176  free_vector(sy);
177
178  if ((f3 = fopen(argv[3], "w")) == NULL) {
179      fprintf(stderr, "Can't open %s\n", argv[3]);
180      exit(3);
181  }
182  for (i = 0; i < SIZE; i++)
183      fprintf(f3, "%f \n", u[i]);
184  fclose(f3);
185
186  return 0;
187 }

```

```

188
189 double menseki(double x0, double x1, double x2,
190               double y0, double y1, double y2)
191 {
192     double s;
193     s = 0.5 * ((x1 - x0) * (y2 - y0) - (y1 - y0) * (x2 - x0));
194     return s;
195 }
196
197 void make(int T, vector ab, vector ab0, int nband, int nb, int SIZE,
198          vector u, vector v, double nu, double rho,
199          vector x, vector y, int **num, int *cum,
200          vector fx, vector fy)
201 {
202     static int first = 1;
203     int i, j, k, l;
204     double a0, a1, a2, b0, b1, b2, c0, c1, c2, nurho, S, D;
205     matrix a, b, c, d;
206     matrix kappa[6], lambda[6];
207
208     if (first)
209         for (i = 0; i < nb * SIZE; i++)
210             ab0[i] = 0.0;
211     for (i = 0; i < nb * SIZE; i++)
212         ab[i] = 0.0;
213
214     double kappa0[6][6][6]
215         = {{{{78, 0, 0, 0, 24, 24}, {-9, 0, 0, 0, 4, -16},
216             {-9, 0, 0, 0, -16, 4}, {12, 0,0, 0, -48, -48},
217             {48, 0, 0, 0, -32, -16}, {48, 0, 0, 0, -16, -32}},
218            {{{-9, 0, 0, 0, 4, -16}, {-18, 0, 0, 0, 24, 120},
219             {11, 0, 0, 0, -16, -16}, {-16, 0, 0, 0, -32, 48},
220             {-20, 0, 0,0, -48, -16}, {-32, 0, 0, 0, -16, 48}},
221            {{{-9, 0, 0, 0, -16, 4}, {11, 0, 0, 0, -16, -16},
222             {-18, 0, 0, 0, 120, 24}, {-16, 0, 0, 0, 48, -32},
223             {-32, 0, 0, 0, 48, -16}, {-20, 0, 0, 0, -16, -48}},
224            {{{12, 0, 0, 0, -48, -48}, {-16, 0, 0, 0, -32, 48},
225             {-16, 0, 0, 0, 48, -32}, {-96, 0,0, 0, 384, 384},
226             {16, 0, 0, 0, 192, 128}, {16, 0, 0, 0, 128, 192}},
227            {{{48, 0, 0, 0, -32, -16}, {-20, 0, 0, 0, -48, -16},
228             {-32, 0, 0, 0, 48, -16}, {16, 0, 0, 0, 192, 128},
229             {160, 0, 0, 0, 384, 128}, {80, 0, 0, 0, 128, 128}},
230            {{{48, 0, 0, 0, -16, -32}, {-32, 0, 0, 0, -16, 48},
231             {-20, 0, 0, 0, -16, -48}, {16, 0, 0, 0, 128, 192},
232             {80, 0, 0, 0, 128, 128}, {160, 0, 0, 0, 128, 384}}}}};
233
234     double kappa1[6][6][6]
235         = {{{{0, -18, 0, 24, 0, 120}, {0, -9, 0, 4, 0, -16},
236             {0, 11, 0, -16, 0, -16}, {0, -20, 0, -48, 0, -16},
237             {0, -16, 0, -32, 0, 48}, {0, -32, 0, -16, 0, 48}},
238            {{{0, -9, 0, 4, 0, -16}, {0, 78, 0, 24, 0, 24},
239             {0, -9, 0, -16, 0, 4}, {0, 48, 0, -32, 0, -16},
240             {0, 12, 0, -48, 0, -48}, {0, 48, 0, -16, 0, -32}},
241            {{{0, 11, 0, -16, 0, -16}, {0, -9, 0, -16, 0, 4},
242             {0, -18, 0, 120, 0, 24}, {0, -32, 0, 48, 0, -16},
243             {0, -16, 0, 48, 0, -32}, {0, -20, 0, -16, 0, -48}},
244            {{{0, -20, 0, -48, 0, -16}, {0, 48, 0, -32, 0, -16},
245             {0, -32, 0, 48, 0, -16}, {0, 160, 0, 384, 0, 128},

```

```

246     {0, 16, 0, 192, 0, 128}, {0, 80, 0, 128, 0, 128}},
247     {{0, -16, 0, -32, 0, 48}, {0, 12, 0, -48, 0, -48},
248     {0, -16, 0, 48, 0, -32}, {0, 16, 0, 192, 0, 128},
249     {0, -96, 0, 384, 0, 384}, {0, 16, 0, 128, 0, 192}},
250     {{0, -32, 0, -16, 0, 48}, {0, 48, 0, -16, 0, -32},
251     {0, -20, 0, -16, 0, -48}, {0, 80, 0, 128, 0, 128},
252     {0, 16, 0, 128, 0, 192}, {0, 160, 0, 128, 0, 384}}};
253
254 double kappa2[6][6][6]
255     = {{{0, 0, -18, 24, 120, 0}, {0, 0, 11, -16, -16, 0},
256     {0, 0, -9, 4, -16, 0}, {0, 0, -20, -48, -16, 0},
257     {0, 0, -32, -16, 48, 0}, {0, 0, -16, -32, 48, 0}},
258     {{0, 0, 11, -16, -16, 0}, {0, 0, -18, 120, 24, 0},
259     {0, 0, -9, -16, 4, 0}, {0, 0, -32, 48, -16, 0},
260     {0, 0, -20, -16, -48, 0}, {0, 0, -16, 48, -32, 0}},
261     {{0, 0, -9, 4, -16, 0}, {0, 0, -9, -16, 4, 0},
262     {0, 0, 78, 24, 24, 0}, {0, 0, 48, -32, -16, 0},
263     {0, 0, 48, -16, -32, 0}, {0, 0, 12, -48, -48, 0}},
264     {{0, 0, -20, -48, -16, 0}, {0, 0, -32, 48, -16, 0},
265     {0, 0, 48, -32, -16, 0}, {0, 0, 160, 384, 128, 0},
266     {0, 0, 80, 128, 128, 0}, {0, 0, 16, 192, 128, 0}},
267     {{0, 0, -32, -16, 48, 0}, {0, 0, -20, -16, -48, 0},
268     {0, 0, 48, -16, -32, 0}, {0, 0, 80, 128, 128, 0},
269     {0, 0, 160, 128, 384, 0}, {0, 0, 16, 128, 192, 0}},
270     {{0, 0, -16, -32, 48, 0}, {0, 0, -16, 48, -32, 0},
271     {0, 0, 12, -48, -48, 0}, {0, 0, 16, 192, 128, 0},
272     {0, 0, 16, 128, 192, 0}, {0, 0, -96, 384, 384, 0}}};
273
274 /* 変数の用意 */
275     a = new_matrix(6, 6);
276     b = new_matrix(6, 3);
277     c = new_matrix(6, 3);
278     d = new_matrix(6, 6);
279
280     for (i = 0; i < 6; i++) {
281         kappa[i] = new_matrix(6, 6);
282         lambda[i] = new_matrix(6, 6);
283     }
284
285 /* A, B, C, D をクリア */
286     for (i = 0; i < 6; i++){
287         for (j = 0; j < 6; j++)
288             a[i][j] = d[i][j] = 0;
289     }
290
291     for (i = 0; i < 6; i++){
292         for (j = 0; j < 3; j++)
293             b[i][j] = c[i][j] = 0;
294     }
295
296     nurho = nu * rho;
297
298     for (l = 0; l < T; l++) { /* 各要素について */
299         int n0, n1, n2;
300         n0 = num[l][0];
301         n1 = num[l][1];
302         n2 = num[l][2];
303         S = menseki(x[n0], x[n1], x[n2], y[n0], y[n1], y[n2]);

```



```

304     if (S < 0.0) {
305         printf("第 %d 要素の面積が負になっています!\n", l);
306         exit(0);
307     }
308     D = 2 * S;
309
310     a0 = (y[n1] - y[n2]) / D;
311     a1 = (y[n2] - y[n0]) / D;
312     a2 = (y[n0] - y[n1]) / D;
313     b0 = (x[n2] - x[n1]) / D;
314     b1 = (x[n0] - x[n2]) / D;
315     b2 = (x[n1] - x[n0]) / D;
316
317     c0 = a1 * a2 + b1 * b2;
318     c1 = a2 * a0 + b2 * b0;
319     c2 = a0 * a1 + b0 * b1;
320
321     a[0][0] = 3 * (a0 * a0 + b0 * b0);
322     a[1][1] = 3 * (a1 * a1 + b1 * b1);
323     a[2][2] = 3 * (a2 * a2 + b2 * b2);
324     a[0][1] = -c2;
325     a[0][2] = -c1;
326     a[1][2] = -c0;
327     a[0][3] = a[1][4] = a[2][5] = 0;
328     a[1][3] = a[2][3] = 4 * c0;
329     a[0][4] = a[2][4] = 4 * c1;
330     a[0][5] = a[1][5] = 4 * c2;
331     a[3][3] = a[4][4] = a[5][5] = -8 * (c0 + c1 + c2);
332     a[3][4] = 8 * c2;
333     a[3][5] = 8 * c1;
334     a[4][5] = 8 * c0;
335
336     for(i = 0; i < 6; i++)
337         for(j = 0; j < i; j++)
338             a[i][j] = a[j][i];
339
340     for(i = 0; i < 6; i++)
341         for(j = 0; j < 6; j++)
342             a[i][j] = (S / 3.0) * a[i][j];
343
344     b[0][0] = a0;
345     b[1][1] = a1;
346     b[2][2] = a2;
347     b[0][1] = b[0][2] = b[1][0] = b[1][2] = b[2][0] = b[2][1] = 0;
348     b[3][0] = a1 + a2;
349     b[4][1] = a2 + a0;
350     b[5][2] = a0 + a1;
351     b[3][1] = a1 + 2 * a2;
352     b[3][2] = 2 * a1 + a2;
353     b[4][0] = 2 * a2 + a0;
354     b[4][2] = a2 + 2 * a0;
355     b[5][0] = a0 + 2 * a1;
356     b[5][1] = 2 * a0 + a1;
357
358     for(i = 0; i < 6; i++)
359         for(j = 0; j < 3; j++)
360             b[i][j] = (S / 3.0) * b[i][j];
361

```

```

362     c[0][0] = b0;
363     c[1][1] = b1;
364     c[2][2] = b2;
365     c[0][1] = c[0][2] = c[1][0] = c[1][2] = c[2][0] = c[2][1] = 0;
366     c[3][0] = b1 + b2;
367     c[4][1] = b2 + b0;
368     c[5][2] = b0 + b1;
369     c[3][1] = b1 + 2 * b2;
370     c[3][2] = 2 * b1 + b2;
371     c[4][0] = 2 * b2 + b0;
372     c[4][2] = b2 + 2 * b0;
373     c[5][0] = b0 + 2 * b1;
374     c[5][1] = 2 * b0 + b1;
375
376     for(i = 0; i < 6; i++)
377         for(j = 0; j < 3; j++)
378             c[i][j] = (S / 3.0) * c[i][j];
379
380     d[0][0] = d[1][1] = d[2][2] = 6;
381     d[0][1] = d[0][2] = d[1][2] = -1;
382     d[0][3] = d[1][4] = d[2][5] = -4;
383     d[0][4] = d[0][5] = d[1][3] = d[1][5] = d[2][3] = d[2][4] = 0;
384     d[3][3] = d[4][4] = d[5][5] = 32;
385     d[3][4] = d[3][5] = d[4][5] = 16;
386
387     for(i = 0; i < 6; i++)
388         for(j = 0; j < i; j++)
389             d[i][j] = d[j][i];
390
391     for(i = 0; i < 6; i++)
392         for(j = 0; j < 6; j++)
393             d[i][j] = (S / 180.0) * d[i][j];
394
395     for(i = 0; i < 6; i++){
396         for(j = 0; j < 6; j++){
397             for(k = 0; k < 6; k++){
398                 kappa[i][j][k] = (a0 * kappa0[i][j][k] +
399                                     a1 * kappa1[i][j][k] +
400                                     a2 * kappa2[i][j][k]) * S / 1260;
401                 lambda[i][j][k] = (b0 * kappa0[i][j][k] +
402                                     b1 * kappa1[i][j][k] +
403                                     b2 * kappa2[i][j][k]) * S / 1260;
404             }
405         }
406     }
407
408     /* 直接剛性法 */
409     /* 線形項について (一部 Stokes と共通) */
410
411     if(first){
412     for(i = 0; i < 6; i++) {
413         int I = cum[num[1][i]];
414         for (j = 0; j < 6; j++) {
415             int J = cum[num[1][j]];
416             A0(I, J) += nurho * a[i][j];
417             A0(I+1, J+1) += nurho * a[i][j];
418         }
419     }

```

```

420         for (j = 0; j < 3; j++) {
421             int J = cum[num[l][j]];
422             A0(I, J+2) -= b[i][j];
423             A0(I+1, J+2) -= c[i][j];
424         }
425     }
426
427     for(i = 0; i < 3; i++) {
428         int I = cum[num[l][i]];
429         for (j = 0; j < 6; j++) {
430             int J = cum[num[l][j]];
431             A0(I+2, J) -= b[j][i];
432             A0(I+2, J+1) -= c[j][i];
433         }
434     }
435 }
436
437 for(i = 0; i < 6; i++) {
438     int I = cum[num[l][i]];
439     for(j = 0; j < 6; j++){
440         v[I] -= rho * d[i][j] * fx[num[l][j]];
441         v[I + 1] -= rho * d[i][j] * fy[num[l][j]];
442     }
443
444     for(j = 0; j < 6; j++){
445         int J = cum[num[l][j]];
446         v[I] += nurho * a[i][j] * u[J];
447         v[I+1] += nurho * a[i][j] * u[J+1];
448     }
449
450     for(j = 0; j < 3; j++){
451         int J = cum[num[l][j]];
452         v[I] -= b[i][j] * u[J+2];
453         v[I+1] -= c[i][j] * u[J+2];
454     }
455 }
456
457 for(i = 0; i < 3; i++) {
458     int I = cum[num[l][i]];
459     for (j = 0; j < 6; j++) {
460         int J = cum[num[l][j]];
461         v[I+2] -= b[j][i] * u[J] + c[j][i] * u[J+1];
462     }
463 }
464
465
466 /* 非線形項について */
467 for (i = 0; i < 6; i++) {
468     int I = cum[num[l][i]];
469     for (j = 0; j < 6; j++) {
470         int J = cum[num[l][j]];
471         for (k = 0; k < 6; k++) {
472             int K = cum[num[l][k]];
473             A(I, J) += rho * (kappa[i][j][k] * u[K] +
474                             kappa[i][k][j] * u[K] +
475                             lambda[i][k][j] * u[K+1]) ;
476             A(I, J+1) += rho * lambda[i][j][k] * u[K];
477             A(I+1, J) += rho * kappa[i][j][k] * u[K+1];

```

```

478             A(I+1, J+1) += rho *(lambda[i][j][k] * u[K+1] +
479                               kappa[i][k][j] * u[K] +
480                               lambda[i][k][j] * u[K+1]);
481         }
482     }
483 }
484
485     for (i = 0; i < 6; i++) {
486         int I = cum[num[l][i]];
487         for (j = 0; j < 6; j++) {
488             int J = cum[num[l][j]];
489             for (k = 0; k < 6; k++) {
490                 int K = cum[num[l][k]];
491                 v[I] += rho * (kappa[i][j][k] * u[J] * u[K]
492                               + lambda[i][j][k] * u[J+1] * u[K]);
493                 v[I+1] += rho * (kappa[i][j][k] * u[J] * u[K+1]
494                                 + lambda[i][j][k] * u[J+1] * u[K+1]);
495             }
496         }
497     }
498 }
499
500     for (i = 0; i < nb * SIZE; i++)
501         ab[i] += ab0[i];
502     first = 0;
503     free_matrix(a);
504     free_matrix(b);
505     free_matrix(c);
506     free_matrix(d);
507     for (i = 0; i < 6; i++) {
508         free_matrix(kappa[i]);
509         free_matrix(lambda[i]);
510     }
511 }
512
513 void change(int n, vector ab, int nband, int nb, int k)
514 {
515     int i, j;
516     for (i = max(0, k-nband); i <= min(n-1, k+nband); i++)
517         A(i, k) = 0.0;
518     for (j = max(0, k-nband); j <= min(n-1, k+nband); j++)
519         A(k, j) = 0.0;
520     A(k, k) = 1.0;
521 }
522
523 void bound(vector ab, int nband, int nb,
524           vector u, int *b, int N, int D, int *cum, int n)
525 {
526     int j, k;
527     for (k = 0; k < D; k++) {
528         j = cum[b[k]];
529         u[j] = 0.0;
530         change(n, ab, nband, nb, j);
531         j++;
532         u[j] = 0.0;
533         change(n, ab, nband, nb, j);
534     }
535     if (N == 0) {

```

```

536     change(n, ab, nband, nb, 2);
537     u[2] = 0.0;
538 }
539 }
540
541 void newton(int T,
542             int nband, int nb, ivector ipvt,
543             vector u, vector v, double nu, double rho,
544             vector x, vector y, int **num, int *cum,
545             vector fx, vector fy, int *b, int N, int D, int SIZE)
546 {
547     int i, q;
548     double norm;
549     vector ab, ab0;
550
551     ab0 = new_vector(nb * SIZE);
552     ab = new_vector(nb * SIZE);
553     if (ab0 == NULL || ab == NULL) {
554         fprintf(stderr, "メモリ不足\n");
555         exit(1);
556     }
557
558     for (q = 0; q <= 10; q++) {
559         for (i = 0; i < SIZE; i++)
560             v[i] = 0.0;
561
562         make(T, ab, ab0, nband, nb, SIZE,
563             u, v, nu, rho, x, y, num, cum, fx, fy);
564
565         bound(ab, nband, nb, v, b, N, D, cum, SIZE);
566
567         /* 連立1次方程式を解く */
568         {
569             int nrhs = 1, ldab = nb, ldb = SIZE, info;
570             dgbsv_(&SIZE, &nband, &nband, &nrhs, ab, &ldab, ipvt, v, &ldb, &info);
571             if (info == 0) {
572                 if (verbose)
573                     printf("successful\n");
574             }
575             else if (info > 0)
576                 printf("U is singular\n");
577             else
578                 printf("%d-th argument has illegal value.\n", abs(info));
579         }
580
581         norm = sqrt(innerproduct(SIZE, v, v) / SIZE);
582         printf("||(\Delta u, \Delta v, \Delta p)||=%e\n", norm);
583
584         for (i = 0; i < SIZE; i++)
585             u[i] -= v[i];
586
587         if (norm < 1.0e-8)
588             return;
589     }
590     printf("収束しませんでした \n");
591 }
592
593

```

594  
595  
596  
597  
598  
599  
600

## 6.5 流速のベクトル場を描くプログラム

```
1 /*
2  * fluid3.c --- 工藤氏 作成 (1995年2月)
3  *             プロトタイプ宣言など修正 (mk, 2004/8/31)
4  *             厳密解がわかっているときは厳密解と
5  *             誤差を求められるようにした (福嶋)
6  */
7
8 #include <math.h>
9 #include <stdio.h>
10 #define G_DOUBLE
11 #include <glsc.h>
12 #include "matutil.h"
13
14 void glsc(int, int, int, int **, int *, vector, vector, vector);
15 void domain(int, int **, vector, vector);
16 void flow(int, int *, vector, vector, vector);
17 void arrow(double, double, double, double);
18
19 int main(int argc, char **argv)
20 {
21     FILE *f1, *f2;
22     int i, j, U, T, SIZE;
23     int *cum, **num;
24     vector x, y, u;
25
26     if (argc != 3) {
27         printf("usage: %s <input> <output>\n", argv[0]);
28         exit(0);
29     }
30     if ((f1 = fopen(argv[1], "r")) == NULL) {
31         fprintf(stderr, "Can't open %s", argv[1]);
32         exit(1);
33     }
34     if ((f2 = fopen(argv[2], "r")) == NULL) {
35         fprintf(stderr, "Can't open %s", argv[2]);
36         exit(2);
37     }
38
39     fscanf(f1, "%d %d", &T, &U);
40
41     x = new_vector(U);
42     y = new_vector(U);
43     for (i = 0; i < U; i++) {
44         fscanf(f1, "%lf %lf", &x[i], &y[i]);
45     }
```

```

46
47     num = (int **) malloc(T * sizeof(void *));
48     if (num == NULL)
49         printf("要素と節点の対応用のメモリーが足りません\n");
50     for (i = 0; i < T; i++) {
51         num[i] = (int *) malloc(sizeof(int) * 6);
52         if (num[i] == NULL) {
53             printf("要素と節点の対応用のメモリーが足りません\n");
54         }
55     }
56     for (i = 0; i < T; i++) {
57         for (j = 0; j < 6; j++) {
58             fscanf(f1, "%d", &num[i][j]);
59         }
60     }
61
62     if ((cum = (int *) malloc(sizeof(int) * U)) == NULL) {
63         printf("累積節点番号用のメモリーが足りません \n");
64     }
65     for (i = 0; i < U; i++) {
66         fscanf(f1, "%d", &cum[i]);
67     }
68
69     fscanf(f1, "%d", &SIZE);
70     fclose(f1);
71
72     u = new_vector(SIZE);
73     for (i = 0; i < SIZE; i++) {
74         fscanf(f2, "%lf", &u[i]);
75     }
76     fclose(f2);
77
78     glsc(U, T, SIZE, num, cum, x, y, u);
79     return 0;
80 }
81
82 void glsc(int U, int T, int SIZE, int **num, int *cum,
83         vector x, vector y, vector u)
84 {
85     int i;
86     double left, right, top, bottom, widx, widy, Lx, Ly;
87
88     left = right = x[0];
89     top = bottom = y[0];
90
91     for (i = 1; i < U; i++) {
92         if (left > x[i])
93             left = x[i];
94         else if (right < x[i])
95             right = x[i];
96         if (top < y[i])
97             top = y[i];
98         else if (bottom > y[i])
99             bottom = y[i];
100    }
101
102    widx = fabs(left) + fabs(right);
103    widy = fabs(top) + fabs(bottom);

```

```

104
105     left -= 0.1 * widx;
106     right += 0.1 * widx;
107     top += 0.1 * widy;
108     bottom -= 0.1 * widy;
109
110     widx = fabs(left) + fabs(right);
111     widy = fabs(top) + fabs(bottom);
112
113     if (widx >= widy) {
114         Lx = 150.0;
115         Ly = Lx * widy / widx;
116     } else {
117         Ly = 150.0;
118         Lx = Ly * widx / widy;
119     }
120
121     g_init("Kudo", Lx + 10.0, Ly + 10.0);
122     g_device(3);
123     g_def_scale(0, left, right, bottom, top, 5.0, 5.0, Lx, Ly);
124     g_cls();
125     g_sel_scale(0);
126
127     printf("例えば 0 6 1 0.3\n");
128     printf("色番号 : 色 \n");
129     printf("0 : 黒  1 : 赤  2 : 緑  3 : 青 \n");
130     printf("4 : 紫  5 : 黄  6 : 水  7 : 白 \n");
131 #ifdef ORIGINAL
132     printf("背景の色番号は? ");
133     scanf("%d", &i);
134 #else
135     i = 6;
136 #endif
137     g_line_color(i);
138     g_area_color(i);
139     g_box(left, right, bottom, top, 1, 1);
140
141     domain(T, num, x, y);
142     flow(U, cum, x, y, u);
143
144     g_sleep(-1.0);
145     g_term();
146 }
147
148 void domain(int T, int **num, vector x, vector y)
149 {
150     int i, j;
151     double zx[3], zy[3];
152
153 #ifdef ORIGINAL
154     printf("領域の色番号は? ");
155     scanf("%d", &i);
156 #else
157     i = 7;
158 #endif
159
160     g_def_line(0, i, 0, 4);
161     g_sel_line(0);

```



```

162     g_area_color(i);
163
164     for (i = 0; i < T; i++) {
165         for (j = 0; j < 3; j++) {
166             zx[j] = x[num[i][j]];
167             zy[j] = y[num[i][j]];
168         }
169         g_polygon(zx, zy, 3, 0, 1);
170     }
171 }
172
173 double norm(double x, double y)
174 {
175     return sqrt(x*x+y*y);
176 }
177
178 void flow(int U, int *cum, vector x, vector y, vector u)
179 {
180     int i, kind, m;
181     double d , r ,theta;
182     vector ux, uy;
183     double c1, c2, mu, nu, omega1, omega2, exponent, bunbo, ur, ut, R1, R2;
184     double error, totalerror, maxerror;
185
186     ux = new_vector(U);
187     uy = new_vector(U);
188
189 #ifdef ORIGINAL
190     printf("↑の色番号は? ");
191     scanf("%d", &i);
192 #else
193     i = 0;
194 #endif
195     g_def_line(0, i, 0, 0);
196     g_sel_line(0);
197
198     printf("↑の長さは? ");
199     scanf("%lf", &d);
200
201     R1 = 0.5; R2 = 1.0;
202     mu = 1.0; nu = 1.0;
203     omega1 = 1.0; omega2 = 1.0; exponent = 2 + mu / nu;
204     bunbo = pow(R2, exponent) - pow(R1, exponent);
205     c1 = (omega1 * R1 * R1 * pow(R2, exponent)
206         - omega2 * R2 * R2 * pow(R1, exponent))/bunbo;
207     c2 = (omega2 * R2 * R2 - omega1 * R1 * R1) / bunbo;
208     printf("c1=%g, c2=%g\n", c1, c2);
209     /* 動径方向 M=5, 偏角方向 N=24 をさらに m 分割したときの m を求める */
210     /* 実際は U = 2M'(2N' - 1) に M'=mM, N'=mN を代入して m について解いた */
211     m = (12 + sqrt(36 + 30 * U)) / 120;
212     for (i = 0; i < U; i++) {
213         r = sqrt(x[i]*x[i] + y[i]*y[i]);
214         theta = atan2(y[i],x[i]);
215         ur = mu / r; ut = (c1 / r + c2 * pow(r, 1 + mu / nu) );
216         ux[i] = ur * cos(theta) - ut * sin(theta);
217         uy[i] = ur * sin(theta) + ut * cos(theta);
218     }
219     totalerror = 0.0; maxerror = 0;

```

```

220 printf("何を書きますか? (0:厳密解, 1:近似解, 2:誤差)");
221 scanf("%d", &kind);
222 for (i = 0; i < U; i++) {
223     /* 分割が大きくなっても平気なように飛び飛びの点で速度を表示させる */
224     if (i % m == 0 && (i / (48 * m)) % (2 * m) == 0){
225         if (kind == 2)
226             arrow(x[i], y[i], x[i] + d * (u[cum[i]] - ux[i]),
227                 y[i] + d * (u[cum[i] + 1] - uy[i]));
228         else if (kind == 0)
229             arrow(x[i], y[i], x[i] + d * ux[i], y[i] + d * uy[i]);
230         else
231             arrow(x[i], y[i], x[i] + d * u[cum[i]], y[i] + d * u[cum[i]+1]);
232     }
233
234     /* 境界 Gamma2 , すなわち外周上においても速度を表示させる. */
235     else if (i % m == 0 && i / (48 * m * (10 * m - 2)) == 1){
236         if (kind == 2)
237             arrow(x[i], y[i], x[i] + d * (u[cum[i]] - ux[i]),
238                 y[i] + d * (u[cum[i] + 1] - uy[i]));
239         else if (kind == 0)
240             arrow(x[i], y[i], x[i] + d * ux[i], y[i] + d * uy[i]);
241         else
242             arrow(x[i], y[i], x[i] + d * u[cum[i]], y[i] + d * u[cum[i]+1]);
243     }
244
245     error = norm(u[cum[i]]-ux[i],u[cum[i]+1]-uy[i]);
246     totalerror += error;
247     if (error > maxerror)
248         maxerror = error;
249 }
250 printf("totalerror=%e\n", sqrt(totalerror));
251 printf("max error=%e\n", maxerror);
252 }
253
254 void arrow(double x1, double y1, double x2, double y2)
255 {
256     double x3, y3, x4, y4;
257
258     x3 = 0.1 * ((x1 + 9.0 * x2) + (y2 - y1));
259     x4 = 0.1 * ((x1 + 9.0 * x2) - (y2 - y1));
260     y3 = 0.1 * ((y1 + 9.0 * y2) - (x2 - x1));
261     y4 = 0.1 * ((y1 + 9.0 * y2) + (x2 - x1));
262
263     g_move(x1, y1);
264     g_plot(x2, y2);
265     g_plot(x3, y3);
266     g_move(x2, y2);
267     g_plot(x4, y4);
268 }
269
270
271
272
273
274
275
276
277

```

278  
279  
280

## 6.6 等圧線を描くプログラム

```
1  /*
2  * contour.c ---工藤プログラムのフォーマットのデータを読んで圧力場を描く
3  *             色付けで高低差がわかるように改良（福島）
4  */
5
6  #include <math.h>
7  #include <stdio.h>
8  #define G_DOUBLE
9  #include <glsc.h>
10 #include "matutil.h"
11
12 void glsc(int, int, int, int **, int *, vector, vector, vector);
13 void domain(int, int **, vector, vector);
14 void flow(int, int *, vector, vector, vector);
15 void arrow(double, double, double, double);
16 void draw_contour(int, int, int, int **, int *, vector, vector, vector);
17 void open_window(int, int, int, int **, int *, vector, vector, vector);
18 void draw_elements(int, int, int, int **, int *, vector, vector, vector);
19 void celldraw(double, double, double,
20              double, double, double,
21              double, double, double,
22              double);
23 void line_level(int ll);
24
25 int main(int argc, char **argv)
26 {
27     FILE *f1, *f2;
28     int i, j, U, T, SIZE;
29     int *cum, **num;
30     vector x, y, u;
31
32     if (argc != 3) {
33 printf("usage: %s <input> <output>\n", argv[0]);
34 exit(0);
35     }
36     if ((f1 = fopen(argv[1], "r")) == NULL) {
37 fprintf(stderr, "Can't open %s", argv[1]);
38 exit(1);
39     }
40     if ((f2 = fopen(argv[2], "r")) == NULL) {
41 fprintf(stderr, "Can't open %s", argv[2]);
42 exit(2);
43     }
44
45     fscanf(f1, "%d %d", &T, &U);
46
47     x = new_vector(U);
48     y = new_vector(U);
49     if (x == NULL || y == NULL) {
```

```

50     fprintf(stderr, "\n");
51     exit(1);
52 }
53 for (i = 0; i < U; i++) {
54 fscanf(f1, "%lf %lf", &x[i], &y[i]);
55 }
56
57     num = malloc(T * sizeof(void *));
58     if (num == NULL)
59 printf("要素と節点の対応用のメモリーが足りません\n");
60     for (i = 0; i < T; i++) {
61 num[i] = malloc(sizeof(int) * 6);
62 if (num[i] == NULL) {
63     fprintf(stderr, "要素と節点の対応用のメモリーが足りません\n");
64     exit(1);
65 }
66     }
67     for (i = 0; i < T; i++)
68 for (j = 0; j < 6; j++)
69     fscanf(f1, "%d", &num[i][j]);
70
71     if ((cum = malloc(sizeof(int) * U)) == NULL) {
72 fprintf(stderr, "累積節点番号用のメモリーが足りません \n");
73 exit(1);
74     }
75     for (i = 0; i < U; i++)
76 fscanf(f1, "%d", &cum[i]);
77
78     fscanf(f1, "%d", &SIZE);
79     fclose(f1);
80
81     u = new_vector(SIZE);
82     if (u == NULL) {
83 fprintf(stderr, "流速・圧力のメモリーが足りません \n");
84 exit(1);
85     }
86     for (i = 0; i < SIZE; i++)
87 fscanf(f2, "%lf", &u[i]);
88     fclose(f2);
89
90     open_window(U, T, SIZE, num, cum, x, y, u);
91
92 #ifdef OLD
93     glsc(U, T, SIZE, num, cum, x, y, u);
94 #else
95     /* draw_elements(U, T, SIZE, num, cum, x, y, u); */
96     draw_contour(U, T, SIZE, num, cum, x, y, u);
97     g_circle(0.0, 0.0, 1.0, G_YES, G_NO);
98     g_circle(0.0, 0.0, 0.5, G_YES, G_NO);
99 #endif
100
101     g_sleep(-1.0);
102     g_term();
103
104     return 0;
105 }
106
107 void open_window(int U, int T, int SIZE, int **num, int *cum,

```

```

108 vector x, vector y, vector u)
109 {
110     int i;
111     double left, right, top, bottom, widx, widy, Lx, Ly;
112     left = right = x[0];
113     top = bottom = y[0];
114
115     for (i = 1; i < U; i++) {
116         if (left > x[i])
117             left = x[i];
118         else if (right < x[i])
119             right = x[i];
120         if (top < y[i])
121             top = y[i];
122         else if (bottom > y[i])
123             bottom = y[i];
124     }
125     printf("left=%g,right=%g\n",left,right);
126     printf("bottom=%g,top=%g\n",bottom,top);
127
128     widx = fabs(left) + fabs(right);
129     widy = fabs(top) + fabs(bottom);
130
131     left -= 0.1 * widx;
132     right += 0.1 * widx;
133     top += 0.1 * widy;
134     bottom -= 0.1 * widy;
135
136     widx = fabs(left) + fabs(right);
137     widy = fabs(top) + fabs(bottom);
138
139     if (widx >= widy) {
140         Lx = 150.0;
141         Ly = Lx * widy / widx;
142     }
143     else {
144         Ly = 150.0;
145         Lx = Ly * widx / widy;
146     }
147     printf("Lx=%g,Ly=%g\n",Lx,Ly);
148     g_init("Kudo", Lx + 10.0, Ly + 10.0);
149     g_device(3);
150     g_def_scale(0, left, right, bottom, top, 5.0, 5.0, Lx, Ly);
151     g_cls();
152     g_sel_scale(0);
153
154     printf("例えば 0 6 1 0.3\n");
155     printf("色番号 : 色 \n");
156     printf("0 : 黒   1 : 赤   2 : 緑   3 : 青 \n");
157     printf("4 : 紫   5 : 黄   6 : 水   7 : 白 \n");
158
159 #ifdef OLD
160     printf("背景の色番号は? ");
161     scanf("%d", &i);
162 #else
163     printf("背景は白にします\n");
164     i = 7;
165 #endif

```

```

166     g_line_color(i);
167     g_area_color(i);
168     g_box(left, right, bottom, top, 1, 1);
169 }
170
171 void glsc(int U, int T, int SIZE, int **num, int *cum,
172     vector x, vector y, vector u)
173 {
174     int i;
175
176     domain(T, num, x, y);
177     flow(U, cum, x, y, u);
178
179 }
180
181 void domain(int T, int **num, vector x, vector y)
182 {
183     int i, j;
184     double zx[3], zy[3];
185
186     printf("領域の色番号は? ");
187     scanf("%d", &i);
188
189     g_def_line(0, i, 0, 4);
190     g_sel_line(0);
191     g_area_color(i);
192
193     for (i = 0; i < T; i++) {
194     for (j = 0; j < 3; j++) {
195         zx[j] = x[num[i][j]];
196         zy[j] = y[num[i][j]];
197     }
198     g_polygon(zx, zy, 3, 0, 1);
199     }
200 }
201
202 /* 流速ベクトル場を描く */
203 void flow(int U, int *cum, vector x, vector y, vector u)
204 {
205     int i;
206     double d;
207
208     printf("↑の色番号は? ");
209     scanf("%d", &i);
210     g_def_line(0, i, 0, 0);
211     g_sel_line(0);
212
213     printf("↑の長さは? ");
214     scanf("%lf", &d);
215
216     for (i = 0; i < U; i++)
217     arrow(x[i], y[i], x[i] + d * u[cum[i]], y[i] + d * u[cum[i] + 1]);
218 }
219
220 /* 矢印を描く */
221 void arrow(double x1, double y1, double x2, double y2)
222 {
223     double x3, y3, x4, y4;

```

```

224
225     x3 = 0.1 * ((x1 + 9.0 * x2) + (y2 - y1));
226     x4 = 0.1 * ((x1 + 9.0 * x2) - (y2 - y1));
227     y3 = 0.1 * ((y1 + 9.0 * y2) - (x2 - x1));
228     y4 = 0.1 * ((y1 + 9.0 * y2) + (x2 - x1));
229
230     g_move(x1, y1);
231     g_plot(x2, y2);
232     g_plot(x3, y3);
233     g_move(x2, y2);
234     g_plot(x4, y4);
235 }
236
237 /* 三角形要素を描く */
238 void draw_elements(int nnode, int num_element, int num_unknowns,
239     int **num, int *cum,
240     vector x, vector y, vector u)
241 {
242     int i, j, p[3];
243
244     printf("要素を描く\n");
245     g_def_line(0, G_BLACK, 0, G_LINE_DOTS);
246     g_sel_line(0);
247     for (i = 0; i < num_element; i++) {
248         for (j = 0; j < 3; j++)
249             p[j] = num[i][j];
250         g_move(x[p[2]], y[p[2]]);
251         for (j = 0; j < 3; j++)
252             g_plot(x[p[j]], y[p[j]]);
253     }
254 }
255
256 /* 等高線に色付け */
257 void line_level(int ll)
258 {
259     int color, type;
260     if(ll == 0 || ll == 1)
261     {
262         type = ll % 2;
263         g_line_color(3);
264         g_line_type(type);
265         g_line_width(2);
266     }
267
268     if(ll == 2 || ll == 3)
269     {
270         type = ll % 2;
271         g_line_color(6);
272         g_line_type(type);
273         g_line_width(2);
274     }
275
276     if(ll == 4 || ll == 5)
277     {
278         type = ll % 2;
279         g_line_color(2);
280         g_line_type(type);
281         g_line_width(2);

```

```

282     }
283
284     if(l1 == 6 || l1 == 7)
285     {
286         type = l1 % 2;
287         g_line_color(4);
288         g_line_type(type);
289         g_line_width(2);
290     }
291
292     else if(l1 == 8){
293         g_line_color(1);
294         g_line_type(0);
295         g_line_width(2);
296     }
297
298     else if(l1 == 9){
299         g_line_color(G_BLACK);
300         g_line_type(0);
301     }
302 }
303
304 /* 等高線を描く */
305 void draw_contour(int nnode, int num_element, int num_unknowns,
306                 int **num, int *cum,
307                 vector x, vector y, vector u)
308 {
309     int i, j, k, p[3], level;
310     double pmax, pmin, h;
311
312     printf("要素を描く\n");
313     g_def_line(2, G_BLACK, 0, G_LINE_SOLID);
314     g_sel_line(2);
315     pmax = pmin = u[2];
316     for (i = 0; i < num_element; i++) {
317         for (j = 0; j < 3; j++) {
318             k = cum[num[i][j]] + 2;
319             if (u[k] > pmax)
320 pmax = u[k];
321             else if (u[k] < pmin)
322 pmin = u[k];
323         }
324     }
325     printf("最大最小ゲット\n");
326     for (level = 0; level < 10; level++) {
327         h = pmin + level * (pmax - pmin) / 10;
328         for (i = 0; i < num_element; i++) {
329             for (j = 0; j < 3; j++)
330 p[j] = num[i][j];
331             celldraw(x[p[0]], x[p[1]], x[p[2]],
332                   y[p[0]], y[p[1]], y[p[2]],
333                   u[cum[p[0]]+2], u[cum[p[1]]+2], u[cum[p[2]]+2], h);
334         }
335         line_level(level);
336     }
337 }
338
339 /* 一つの三角形内で高さ h の等高線を描く */

```



```

340 void celldraw(double x1, double x2, double x3,
341             double y1, double y2, double y3,
342             double u1, double u2, double u3,
343             double h)
344 {
345     double px[4], py[4];
346     int n;
347     double r;
348
349     n = 0;
350
351     if ((u1-h)*(u2-h)<=0) {
352         n++;
353         if (u1==h) {
354             px[n] = x1;
355             py[n] = y1;
356         }
357         else if (u2==h) {
358             px[n] = x2;
359             py[n] = y2;
360         }
361         else {
362             r = (u1 - h) / (u1 - u2);
363             px[n] = (1 - r) * x1 + r * x2;
364             py[n] = (1 - r) * y1 + r * y2;
365         }
366     }
367     if ((u2-h)*(u3-h)<=0) {
368         n++;
369         if (u2==h) {
370             px[n] = x2;
371             py[n] = y2;
372         }
373         else if (u3==h) {
374             px[n] = x3;
375             py[n] = y3;
376         }
377         else {
378             r = (u2 - h) / (u2 - u3);
379             px[n] = (1 - r) * x2 + r * x3;
380             py[n] = (1 - r) * y2 + r * y3;
381         }
382     }
383     if ((u3-h)*(u1-h)<=0) {
384         n++;
385         if (u3==h) {
386             px[n] = x3;
387             py[n] = y3;
388         }
389         else if (u1==h) {
390             px[n] = x1;
391             py[n] = y1;
392         }
393         else {
394             r = (u3 - h) / (u3 - u1);
395             px[n] = (1 - r) * x3 + r * x1;
396             py[n] = (1 - r) * y3 + r * y1;
397         }

```

```
398     }
399
400     if (n==2) {
401         g_move(px[1], py[1]);
402         g_plot(px[2], py[2]);
403     }
404     else if (n==3) {
405         g_move(px[1], py[1]);
406         g_plot(px[2], py[2]);
407         g_plot(px[3], py[3]);
408         g_plot(px[1], py[1]);
409     }
410 }
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
```

## 謝辞

本論文の作成に限らず、長きにわたりご指導と暖かい励ましを頂きました森本 浩子 教授に深く感謝いたします。また、本年度ご指導を頂きました桂田 祐史 助教授と、元となるプログラムを提供していただいた工藤 丈征 氏に心から御礼申し上げます。

## 参考文献

- [1] 川原 睦人, 有限要素法流体解析, 日科技連 (1985)
- [2] 菊地 文雄, 有限要素法の数理, 培風館 (1994)
- [3] 菊地 文雄, 有限要素法概説 [新訂版], サイエンス社 (1999)
- [4] 工藤 丈征, 有限要素法による Navier-Stokes 方程式の数値解析プログラム取扱説明書, 明治大学大学院 (1995)
- [5] 塩谷 光晴, 定常 Stokes 方程式の解の精度保証付き数値計算, 明治大学大学院修士学位請求論文 (2003)

- [6] 杉原正顯・室田一雄, 数値計算法の数理, 岩波書店 (1994)
- [7] H.Morimoto, A Solution to the Stationary Navier-Stokes Equation under the Boundary Condition with Non-Vanishing Outflow, Memoirs of the Institute of Science and Technology Meiji University Vol.31 No.2(1992)
- [8] H.Morimoto and S.Ukai, Perturbation of the Navier-Stokes flow in an annular domain with the non-vanishing outflow condition, J.Math.Sci. Univ. Tokyo **3**,73-82(1996)
- [9] H.Morimoto, General Outflow Condition for Navier-Stokes Flow, Num. Appl. Anal., 16, 209-224(1998)