定常 Stokes 方程式の有限要素解の事後誤差評価と事前誤差評価

明治大学大学院
理工学研究科 基礎理工学専攻 数学系
福澤 誠人
指導教員 桂田 祐史 助教授

2005 年 2 月 25 日

目 次

1 はじめに

精度保証付き数値計算では、数値計算した解が真の解にどれだけ近いか(精度がどれほどか)を具体的に評価するものだが、不動点定理の条件の成立を数値計算で確認することで、非線形方程式の解の存在を証明することもできる。このように精度保証付き数値計算には二つの意味があるが、今回は前者の意味でのみの精度保証付き数値計算を行う。

従来この精度保証付き数値計算を行うにはたくさんの困難さがあった。当初は区間演算を行うことさえ難しかったが、最近のパーソナルコンピュータやワークステーションでは IEEE754 規格に準拠した CPU が採用されているので、区間演算に必要な丸めのコントロールを比較的簡単にできるようになり、区間演算は容易に実現できる。また基本的アルゴリズムの研究が進み、それに伴って手軽に用いることができるソフトウェアの研究が進んだことも大きい。今回は MATLAB で区間演算を実現するツールボックスである S.M.Rump による INTLAB(`http://www.ti3.tu-harburg.de/~rump/intlab/`) を用いたが、これを用いると MATLAB の通常の計算と同様に簡単に区間演算による計算を行うことができる。

この論文では、中尾充宏、山本野人、渡部善隆による Stokes 問題に対する有限要素解の精度保証の評価法 [1] を元に、事後誤差評価式 (3.10) と事前誤差評価式 (4.5), (4.8) に現れる右辺を評価する数値実験の追試を行い、さらに区間演算を用いて厳密な意味での精度保証付き数値実験を行うことを目標にして取り組んだ。

この論文の概略は次のようなものである。まず第 1 節で扱う問題と関数空間を設定し、第 2 節で弱解の存在を保証する inf-sup 条件に関連した定数の数値評価を与える。第 3 節で事後誤差評価式を導き、第 4 節では事前誤差評価式とその構成法を述べる。第 5 節では Aubin-Nitsche's trick に似た手法を用いて流速に関する L^2 ノルムでの事後誤差評価式と事前誤差評価式を与える。第 6 節では事前誤差評価をする際に必要となる一般化固有値の絶対値の最大の精度保証付きでの評価法について述べる。第 7 節と第 8 節で事前誤差評価と事後誤差評価それぞれの具体的な計算方法と実験結果について述べる。

1.1 Stokes 方程式

次の凸多角形領域 $\Omega \subset \mathbf{R}^2$ における Stokes 方程式の (Dirichlet) 境界値問題を考える。

$$\begin{cases} -\nu\triangle u + \nabla p = f & \text{in } \Omega, \\ \operatorname{div} u = 0 & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega. \end{cases} \tag{1.1}$$

ここで、$\nu > 0$ は粘性係数 (正定数)、$u = [u_1, u_2]^T$ は二次元速度ベクトル、$f = [f_1, f_2]^T$ は単位質量に働く外力密度で滑らかな関数、p は圧力である。

1.2 関数空間

$H^k(\Omega)$ を Ω 上での k 階の Sobolev 空間、$(\cdot,\cdot)$ を $L^2(\Omega)$ での内積、$(\nabla\cdot,\nabla\cdot)$ を $H_0^1(\Omega)$ での内積とする。以下のように関数空間を定義する。

$$\begin{aligned} H_0^1(\Omega) &:= \left\{v \in H^1(\Omega)\ ;\ v = 0 \text{ on } \partial\Omega\right\}, \\ L_0^2(\Omega) &:= \left\{v \in L^2(\Omega)\ ;\ (v,1) = 0\right\}, \\ \mathscr{S} &:= H_0^1(\Omega)^2 \times L_0^2(\Omega). \end{aligned}$$

$L^2(\Omega),\ H_0^1(\Omega)$ のノルムはそれぞれ

$$\begin{aligned} |q|_0 &:= (q,q)^{\frac{1}{2}}, \\ |v|_1 &:= |\nabla v|_0 \end{aligned}$$

とする。また $H^2(\Omega)$ のセミノルムとして

$$|u|_2 := \left(\left| \frac{\partial^2 u}{\partial x^2} \right|_0^2 + 2 \left| \frac{\partial^2 u}{\partial x \partial y} \right|_0^2 + \left| \frac{\partial^2 u}{\partial y^2} \right|_0^2 \right)^{\frac{1}{2}}$$

を用いる。そして関数空間 $V,\ V^\perp$ を以下のように定義する。

$$\begin{aligned} V &:= \left\{v \in H_0^1(\Omega)^2\ ;\ \operatorname{div} v = 0\right\}, \\ V^\perp &:= \left\{v \in H_0^1(\Omega)^2\ ;\ (\nabla v, \nabla w) = 0,\ w \in V\right\}. \end{aligned}$$

最後に、$H^{-1}(\Omega)^2$ を $H_0^1(\Omega)^2$ の双対空間とする。

2 inf-sup 条件の定数の評価

この節では Stokes 方程式の境界値問題 (1.1) を弱形式で書き直し、弱解の存在を保証する inf-sup 条件に関連した定数の数値評価を与える。この定数を用いて、$\mathscr{S}$ のそれぞれの要素に対してノルム不等式を与える。

2.1 変分法的定式化

Stokes 方程式の境界値問題 (1.1) に対する弱形式として

$$\begin{aligned} \text{Find} \quad & [u,p] \in H_0^1(\Omega)^2 \times L^2(\Omega) \quad \text{s.t.} \\ & \nu(\nabla u, \nabla v) - (p, \operatorname{div} v) = (f,v), \quad \forall v \in H_0^1(\Omega)^2, \\ & -(\operatorname{div} u, q) = 0, \qquad \forall q \in L^2(\Omega) \end{aligned} \tag{2.1}$$

が得られる。次のように記号の定義を行う。

$$\begin{cases} X = H_0^1(\Omega)^2,\ Y = L^2(\Omega),\ f \in L^2(\Omega)^2, & \\ a(u,v) := \nu(\nabla u, \nabla v) & (u,v \in V), \\ F(v) := (f,v) & (v \in V), \\ Bv := -\operatorname{div} v (\in W) & (v \in V), \\ g := 0 (\in R(B)) & (R(B) \text{ は } B \text{ の値域}). \end{cases} \tag{2.2}$$

この定式化から次の定理を用いれば (2.1) の解 $[u,p]$ の存在と一意性を得る (ただし p については定数関数分の不定性がある)。

Theorem 2.1
X, Y は実 Hilbert 空間、$a(\cdot,\cdot) : X\times X \to \mathbf{R}$ は有界対称な双 1 次形式で非負、$F(\cdot) : X \to \mathbf{R}$ は有界線形汎関数、$B : X \to Y$ は有界線形作用素で $R(B)$ は Y の閉集合、g は $R(B)$ の与えられた元とする。さらに $a(\cdot,\cdot)$ は $N(B)$ で強圧的 (ここで $N(B)$ は B の零空間)、すなわち正定数 μ が存在して次式が成立すると仮定する。

$$a(v,v) \geq \mu\|v\|_V^2, \quad (\forall v \in N(B)).$$

このとき

$$\begin{aligned} \text{Find} \quad & [u,p] \in X \times Y \quad \text{s.t.} \\ & a(u,v) + (Bv,p)_Y &=& \ F(v), \quad \forall v \in X, \\ & (Bu,q)_Y &=& \ g, \qquad \forall q \in Y \end{aligned}$$

の解は存在し、その 1 つを $\{u,p\} \in X \times Y$ と記すと、u は X で一意に定まり、p は $R(B)$ 内に限れば一意であり、Y では $N(B^*)$ 分だけの付加項の不定性がある、つまり $p',\ p^* \in Y$ を条件を満たす $p \in Y$ とすると、$p' - p^* \in N(B^*)$ となる (ここで $(\cdot,\cdot)_Y$ は Y での内積、また B^* は B の共役作用素)。

この Theorem の証明は菊地 [12] に載っている。□

(2.2) の定式化において $N(B^*)$ は定数関数である。なぜなら、$q \in Y,\ v \in X$ に対し

$$(B^*q, v)_X = (q, Bv)_Y = -(q, \operatorname{div} v)$$

に注意し、さらに $v \in C_0^\infty(\Omega)^2$ (この集合は $X = H_0^1(\Omega)^2$ で稠密) ととれば、$q \in N(B^*)$ すなわち $B^*q = 0$ となるための q に対する条件は、$\nabla q = 0$ となるからである。

Theorem 2.1 において F は X 上の有界線形汎関数であること、g は $R(B)$ に含まれることのみを要請する。さらに $N(B^*)$ は (2.2) の定式化において定数関数であるので、(2.1) の解の一意存在は次の定理に拡張することができる。

Theorem 2.2
$S \in H^{-1}(\Omega)^2,\ R \in (L_0^2(\Omega))'$ とする。このとき、

$$\begin{aligned} \text{Find} \quad & [u,p] \in H_0^1(\Omega)^2 \times L_0^2(\Omega) \quad \text{s.t.} \\ & \nu(\nabla u, \nabla v) - (p, \operatorname{div} v) &=& \ Sv, \quad \forall v \in H_0^1(\Omega)^2, \\ & -(\operatorname{div} u, q) &=& \ Rq, \quad \forall q \in L_0^2(\Omega) \end{aligned} \tag{2.3}$$

の解 $[u,p] \in H_0^1(\Omega)^2 \times L_0^2(\Omega)$ は一意に存在する。

次に $\mathscr{S} \times \mathscr{S}$ 上の双線形形式 $\mathscr{L}$ を次のように定める。

$$\mathscr{L}([u,p],[v,q]) := \nu(\nabla u, \nabla v) - (p, \operatorname{div} v) - (q, \operatorname{div} u), \quad [u,p],\ [v,q] \in \mathscr{S}. \tag{2.4}$$

このとき、(1.1) の変分法的定式化を次で与える。

$$\text{Find} \quad [u,p] \in \mathscr{S} \quad \text{s.t.} \quad \mathscr{L}([u,p],[v,q]) = (f,v), \quad \forall [v,q] \in \mathscr{S}. \tag{2.5}$$

(2.3) において $Sv = (f,v)$, $R = 0$ とした問題と (2.5) の同値性については明らかであるので、(2.5) は Theorem 2.2 を用いると、$\mathscr{S}$ で一意解を持つことがわかる。

また次に述べる定理により、このとき Ω のみに依存する正定数 β_c が存在して

$$\inf_{\substack{[u,p]\in\mathscr{S},\\ [u,p]\neq 0}} \sup_{\substack{[v,q]\in\mathscr{S},\\ [v,q]\neq 0}} \frac{\mathscr{L}([u,p],[v,q])}{(|u|_1+|p|_0)(|v|_1+|q|_0)} \geq \beta_c \tag{2.6}$$

が成り立つ。

Theorem 2.3
X, Y : Hilbert 空間、 $a(\cdot,\cdot) : X \times Y \to \mathbf{R}$ を連続双線形形式とする。このとき方程式

$$\text{Find} \quad x \in X \quad \text{s.t.} \quad a(x,y) = Fy, \quad \forall y \in Y \tag{2.7}$$

が任意の $F \in Y'$ に対して一意解を持つためには a が次の 2 つの条件を満たすことが必要十分である。

$$C_1 := \inf_{x\in X} \sup_{y\in Y} \frac{a(x,y)}{\|x\|_X \|y\|_Y} > 0, \tag{2.8}$$

$$\sup_{x\in X} \frac{a(x,y)}{\|x\|_X} > 0, \quad \forall y \in Y, \quad y \neq 0. \tag{2.9}$$

さらに (2.7) の解 $x \in X$ は、

$$\|x\|_X \leq \frac{\|F\|_{Y'}}{C_1}$$

を満たす。

この Theorem の証明は土屋 [14] に載っている。

この Theorem を適用して (2.6) をみたす正定数 β_c の存在を証明するには、$X = Y = \mathscr{S}$, $Y' = \mathscr{S}'$ とした上で、

$$\forall F \in \mathscr{S}', \quad \exists! S \in H_0^{-1}(\Omega)^2, \quad \exists! R \in (L_0^2(\Omega))' \quad \text{s.t.} \quad F[v,q] = Sv + Rq \quad ([v,q] \in \mathscr{S})$$

となることに注意して Theorem 2.2 を用いればよい。

(2.6) から

$$\sup_{\substack{[v,q]\in\mathscr{S},\\ [v,q]\neq 0}} \frac{\mathscr{L}([u,p],[v,q])}{|v|_1+|q|_0} \geq \beta_c(|u|_1+|p|_0), \quad \forall [u,p] \in \mathscr{S} \tag{2.10}$$

を得る。$[u,p] \in \mathscr{S}$ に対して

$$\delta(u,p) = \sup_{\substack{[v,q]\in\mathscr{S},\\ [v,q]\neq 0}} \frac{\mathscr{L}([u,p],[v,q])}{|v|_1+|q|_0} \tag{2.11}$$

と定義する。この節の残りで $\delta(u,p)$ を用いて $|u|_1$ と $|p|_0$ を評価する (Theorem 2.4)。そのために、次の Lemma 2.1 を用いる。

Lemma 2.1 (Babuška-Aziz の不等式)
Ω を $\mathbf{R}^2$ の有界 Lipschitz 領域とする。このとき、
$\exists\beta > 0, \quad \forall q \in L^2_0(\Omega), \quad \exists! v \in V^\perp \quad$ s.t.

$$\begin{aligned} \operatorname{div} v &= q, && (2.12)\\ |v|_1 &\leq \frac{1}{\beta}|q|_0. && (2.13) \end{aligned}$$

ただし、$\beta > 0$ は Ω に依存する定数である。

この Lemma の証明は Roger Temam [20] を見よ。Ω が滑らかな境界を持つ場合の証明は付録に載せる。

Lemma 2.1 から

$$\inf_{\substack{p \in L^2_0(\Omega),\\ p \neq 0}} \sup_{\substack{u \in H^1_0(\Omega)^2,\\ u \neq 0}} \frac{-(p, \operatorname{div} v)}{|u|_1 |p|_0} \geq \beta \tag{2.14}$$

を得る。(2.14) は (2.5) が $\mathscr{S}$ で一意解を持つことを保証する $\mathscr{S}$ の inf-sup 条件と呼ばれる条件である。もし Ω が一般の有界連結領域ならば、β の上界や下界を評価することは難しい。しかしながら、Ω が星型の場合、この定数 β は次の Horgan の Lemma 2.2 により数値的に決定できる。

Lemma 2.2 (Horgan [7])
2 次元の有界 Lipschitz 領域 Ω は原点に関して星型であり、境界は次式によって極座標で表されるとする。

$$r = f(\theta) \quad \text{on} \quad \partial\Omega.$$

また、

$$\mathscr{F}(\theta) := \left\{ \left[1 + \left(\frac{f'(\theta)}{f(\theta)} \right)^2 \right]^{\frac{1}{2}} + \frac{|f'(\theta)|}{f(\theta)} \right\}^2$$

とする。このとき $\dfrac{1}{\beta} = \sqrt{1 + \max_\theta \mathscr{F}(\theta)}$ とおくと Lemma 2.1 の不等式 (2.13) が成り立つ。

$\Omega = (-1,-1)\times(1,1)$ の場合、

$$f(\theta) = \begin{cases} \dfrac{1}{\cos\theta} & \left(-\dfrac{\pi}{4} \le \theta \le \dfrac{\pi}{4}\right), \\ \dfrac{1}{\sin\theta} & \left(\dfrac{\pi}{4} \le \theta \le \dfrac{3\pi}{4}\right), \\ -\dfrac{1}{\cos\theta} & \left(\dfrac{3\pi}{4} \le \theta \le \dfrac{5\pi}{4}\right), \\ -\dfrac{1}{\sin\theta} & \left(\dfrac{5\pi}{4} \le \theta < \dfrac{7\pi}{4}\right) \end{cases}$$

である。このとき、

$$\mathscr{F}(\theta) = \begin{cases} -1+\dfrac{2}{1+\sin\theta} & \left(-\dfrac{\pi}{4} \le \theta \le 0,\ \pi \le \theta \le \dfrac{5\pi}{4}\right), \\ -1+\dfrac{2}{1-\sin\theta} & \left(0 \le \theta \le \dfrac{\pi}{4},\ \dfrac{3\pi}{4} \le \theta \le \pi\right), \\ -1+\dfrac{2}{1-\cos\theta} & \left(\dfrac{\pi}{4} \le \theta \le \dfrac{\pi}{2},\ \dfrac{3\pi}{2} \le \theta < \dfrac{7\pi}{4}\right), \\ -1+\dfrac{2}{1+\cos\theta} & \left(\dfrac{\pi}{2} \le \theta \le \dfrac{3\pi}{4},\ \dfrac{5\pi}{4} \le \theta \le \dfrac{3\pi}{2}\right) \end{cases}$$

であり、$\theta = -\dfrac{\pi}{4},\ \dfrac{\pi}{4},\ \dfrac{3\pi}{4},\ \dfrac{5\pi}{4}$ のとき $\mathscr{F}(\theta)$ は最大値 $3+2\sqrt{2}$ をとる。よって、

$$\sqrt{1+\max_{\theta}\mathscr{F}(\theta)} = \sqrt{4+2\sqrt{2}}$$

となる。この結果は Ω が一般の正方形領域の場合にも成り立つ。

2.2 ノルム不等式

Lemma 2.1 の定数 β を用いて、次の不等式を得る。

Theorem 2.4
$\forall [u,p] \in \mathscr{S}$ に対して $\delta(u,p)$ を (2.11) によって定める。このとき、次の評価が成り立つ。

$$\begin{aligned} |u|_1 &\le \left(\frac{1}{\nu^2}+\frac{1}{\beta^2}\right)^{\frac{1}{2}} \delta(u,p), \\ |p|_0 &\le \left(\frac{1}{\beta}+\frac{\nu}{\beta^2}\right) \delta(u,p). \end{aligned} \tag{2.15}$$

証明
V は $H_0^1(\Omega)^2$ の閉部分空間であるので、$H_0^1(\Omega)^2 = V \oplus V^\perp$ である。よって、

$$\forall u \in H_0^1(\Omega)^2, \quad \exists! w \in V, \quad \exists! u_0 \in V^\perp, \quad u = w + u_0.$$

$w \neq 0$ とする。$\delta(u,p)$ の定義式 (2.11) で $v = w,\ q = 0$ とすると、

$$\delta(u,p) \geq \frac{\nu(\nabla u, \nabla w) - (p, \operatorname{div} w)}{|w|_1} = \frac{\nu(\nabla w, \nabla w)}{|w|_1} \geq \nu |w|_1 \tag{2.16}$$

を得る。この式は $w = 0$ でも成り立つ。
一方、$u_0 \neq 0$ であるとき $\delta(u,p)$ の定義式で $v = 0$ とすると、

$$\delta(u,p) \geq \frac{-(q, \operatorname{div} u)}{|q|_0}, \quad 0 \neq \forall q \in L_0^2(\Omega).$$

よって、$q = -\operatorname{div} u_0$ とおくと $q \in L_0^2(\Omega)$ で

$$\beta |u_0| \leq |q|_0$$

が成り立つ。このとき、

$$\delta(u,p) \geq |q|_0 \geq \beta |u_0|_1 \tag{2.17}$$

となる。この式は $u_0 = 0$ のときも成り立つ。よって、(2.16), (2.17) を用いると

$$|u|_1^2 = (\nabla(w+u_0), \nabla(w+u_0)) = (\nabla w, \nabla w) + (\nabla u_0, \nabla u_0) = |w|_1^2 + |u_0|_1^2 \leq \left(\frac{1}{\nu^2} + \frac{1}{\beta^2} \right) \delta(u,p)^2$$

を得る。(2.15) の 1 つ目の不等式を示すことができた。
次に (2.15) の 2 つ目の不等式を示す。$\forall p \in L_0^2(\Omega)$ に対し、Lemma 2.1 より $v \in V^\perp$ として

$$\operatorname{div} v = -p, \quad |v|_1 \leq \frac{1}{\beta} |p|_0 \tag{2.18}$$

を満たすようにとれる。$p \neq 0$ とする。$(\nabla u, \nabla v) = (\nabla u_0, \nabla v)$ より、

$$\delta(u,p) \geq \frac{\nu(\nabla u_0, \nabla v) - (q, \operatorname{div} u_0) + |p|_0^2}{|v|_1 + |q|_0}, \quad \forall q \in L_0^2(\Omega).$$

まず $u_0 \neq 0$ のとき、$q \in L_0^2(\Omega)$ を以下のように定める。

$$q := K \operatorname{div} u_0, \quad K := \frac{\nu(\nabla u_0, \nabla v)}{|\operatorname{div} u_0|_0^2}.$$

上式より、

$$\nu(\nabla u_0, \nabla v) - (q, \operatorname{div} u_0) = \nu(\nabla u_0, \nabla v) - K(\operatorname{div} u_0, \operatorname{div} u_0) = 0. \tag{2.19}$$

さらに、(2.13) より

$$|u_0|_1 \leq \frac{1}{\beta} |\operatorname{div} u_0|_0. \tag{2.20}$$

したがって、

$$|q|_0 = K|\operatorname{div} u_0|_0 = \frac{\nu(\nabla u_0, \nabla v)}{|\operatorname{div} u_0|_0} \le \frac{\nu|u_0|_1|v|_1}{|\operatorname{div} u_0|_0}.$$

よって、(2.18), (2.20) より、

$$|q|_0 \le \frac{\nu}{\beta^2}|p|_0. \tag{2.21}$$

(2.19), (2.21) は $q = 0$ ととることにより $u_0 = 0$ の場合も成り立つ。ゆえに、

$$\delta(u,p) \ge \frac{\nu(\nabla u_0, \nabla v) - (q, \operatorname{div} u_0) + |p|_0^2}{|v|_1 + |q|_1} \ge \frac{|p|_0^2}{\dfrac{1}{\beta}|p|_0 + \dfrac{\nu}{\beta^2}|p|_0} = \left(\frac{1}{\beta} + \frac{\nu}{\beta^2}\right)^{-1}|p|_0.$$

上式は $p = 0$ のときも成り立つ。よって (2.15) の 2 つ目の不等式も示すことができた。□

3 事後誤差評価

この節では流速と圧力を近似する有限要素空間を導入する。そして Theorem 2.4 を用いて Stokes 方程式の事後誤差限界を示す。

3.1 有限要素部分空間

$\mathscr{T}_h$ を $\Omega \subset \mathbf{R}^2$ のスケールパラメータ $h > 0$ に依存する三角形または四角形からなる分割族とする。また、$X_h \subset H_0^1(\Omega) \cap C(\overline{\Omega})$ を流速 u を近似する有限要素部分空間、$Y_h \subset L_0^2(\Omega) \cap C(\overline{\Omega}) \cap H^1(\Omega)$ を圧力 p を近似する有限要素部分空間、X_h^* を X_h の基底関数と $\partial\Omega$ 上の節点に対応する基底関数で張られる空間、$S_h := X_h^2$ とする。

注意　中尾、山本、渡辺 [1] では $Y_h \subset H^1(\Omega)$ と書かれていなかったが、暗に仮定されていて、数値実験で採用された Y_h はこの性質を満たしている。

(2.5) の有限要素解は次で定義される。

$$\text{Find}\ \ [u_h, p_h] \in S_h \times Y_h\ \ \text{s.t.}\ \ \mathscr{L}([u_h, p_h], [v_h, q_h]) = (f, v_h),\ \ \forall [v_h, q_h] \in S_h \times Y_h. \tag{3.1}$$

山本、中尾 [8] で提案された事後処理の手順を示す。

$$X_h \subsetneq X_h^* \subset H^1(\Omega), \quad X_h \ne X_h^*$$

であり X_h^* は $H^1(\Omega)$ の要素を近似する能力を持つ。また、L^2-射影 $P_0 : L^2(\Omega) \to X_h$,　L^2-射影 $\hat{P}_0 : L^2(\Omega) \to X_h^*$,　H_0^1-射影 $P_1 : H_0^1(\Omega) \to X_h$ を以下のように定める。

$$\begin{aligned}(v - P_0 v, \phi) &= 0, \quad \forall \phi \in X_h,\\ (v - \hat{P}_0 v, \phi) &= 0, \quad \forall \phi \in X_h^*,\\ (\nabla(v - P_1 v), \nabla\phi) &= 0, \quad \forall \phi \in X_h.\end{aligned}$$

また、$w_h \in X_h$ に対して、$\overline{\nabla} w_h \in (X_h^*)^2$ と $\overline{\triangle} w_h \in L^2(\Omega)$ を

$$\begin{aligned}\overline{\nabla} w_h &:= \left(\hat{P}_0 \frac{\partial w_h}{\partial x}, \hat{P}_0 \frac{\partial w_h}{\partial y}\right),\\ \overline{\triangle} w_h &:= \operatorname{div} \overline{\nabla} w_h\end{aligned}$$

により定める。$\forall v_h \in S_h$ に対して、

$$
\begin{aligned}
(-\overline{\triangle} v_h, \phi) &= (\overline{\nabla} v_h, \nabla \phi), \quad \forall \phi \in H_0^1(\Omega)^2, && (3.2)\\
|\overline{\nabla} v_h - \nabla v_h|_0 &= \inf_{w_h \in (x_h^*)^2 \times (x_h^*)^2} |w_h - \nabla v_h|_0 && (3.3)
\end{aligned}
$$

が成り立つ。以下、上式を示す。まず、(3.2) については、$v_h = [v_{h1}, v_{h2}] \in S_h,\ \phi = [\phi_1, \phi_2] \in H_0^1(\Omega)^2$ として

$$
\begin{aligned}
(-\overline{\triangle} v_h, \phi) &= (-\overline{\triangle} v_{h1}, \phi_1) + (-\overline{\triangle} v_{h2}, \phi_2)\\
&= (-\operatorname{div} \overline{\nabla} v_{h1}, \phi_1) + (-\operatorname{div} \overline{\nabla} v_{h2}, \phi_2)\\
&= (\overline{\nabla} v_{h1}, \nabla \phi_1) + (\overline{\nabla} v_{h2}, \nabla \phi_2)\\
&= (\overline{\nabla} v_h, \nabla \phi).
\end{aligned}
$$

よって (3.2) は示せた。次に (3.3) については、$v_h = [v_{h1}, v_{h2}] \in S_h$ として

$$
|\overline{\nabla} v_h - \nabla v_h|_0^2 = |\overline{\nabla} v_{h1} - \nabla v_{h1}|_0^2 + |\overline{\nabla} v_{h2} - \nabla v_{h2}|_0^2.
$$

ここで、

$$
\begin{aligned}
|\overline{\nabla} v_{h1} - \nabla v_h|_0^2 &= \left| \hat{P}_0 \frac{\partial v_{h1}}{\partial x} - \frac{\partial v_{h1}}{\partial x} \right|_0^2 + \left| \hat{P}_0 \frac{\partial v_{h1}}{\partial y} - \frac{\partial v_{h1}}{\partial y} \right|_0^2\\
&= \inf_{w_{h1_x} \in X_h^*} \left| w_{h1_x} - \frac{\partial v_{h1}}{\partial x} \right|_0^2 + \inf_{w_{h1_y} \in X_h^*} \left| w_{h1_y} - \frac{\partial v_{h1}}{\partial y} \right|_0^2\\
&= \inf_{w_{h1} \in (X_h^*)^2} |w_{h1} - \nabla v_{h1}|_0^2 .
\end{aligned}
$$

同様に、

$$
|\overline{\nabla} v_{h2} - \nabla v_{h2}|_0^2 = \inf_{w_{h2} \in (X_h^*)^2} |w_{h2} - \nabla v_{h2}|_0^2 .
$$

よって、

$$
|\overline{\nabla} v_h - \nabla v_h|_0^2 = \inf_{w_h \in (X_h^*)^2 \times (X_h^*)^2} |w_h - \nabla v_h|_0^2 .
$$

上式で正の平方根をとれば (3.3) を得る。□

ここで、X_h に次の仮定をおく。

Assumption 3.1

$$
\inf_{\xi \in X_h} |v - \xi|_1 \le C_0 h |v|_2, \qquad \forall v \in H_0^1(\Omega) \cap H^2(\Omega) \tag{3.4}
$$

をみたす v, h に依存しない正定数 C_0 が具体的に評価可能である。

この仮定は多くの有限要素空間で成り立つ。X_h が 1 次元の区分 1 次要素の空間では $C_0 = \dfrac{1}{\pi}$ ととれる。また 1 次元の区分 2 次要素のテンソル積として定義される 2 次元矩形要素では、$C_0 = \dfrac{1}{2\pi}$ とれる (中尾、山本、木村 [9])。射影 P_1 については、Assumption 3.1 と Aubin-Nitche's trick から、

任意の $v \in H_0^1(\Omega)$ に対して、

$$|v - P_1 v|_1 \leq |v|_1, \tag{3.5}$$
$$|v - P_1 v|_0 \leq C_0 h |v|_1 \tag{3.6}$$

が成り立つ。(3.5) はピタゴラスの定理より明らかに成り立つ。以下 (3.6) を示す。

$v \in H_0^1(\Omega)$ とする。ここで $g = v - P_1 v$ とおく。$g \in L^2(\Omega)$ である。次の問題を考える。

$$\text{Find} \quad \psi \in H_0^1(\Omega) \quad \text{s.t.} \quad (\nabla\psi, \nabla\varphi) = (g, \varphi), \quad \forall \varphi \in H_0^1(\Omega).$$

Riez の定理から、この問題の解 ψ は存在する。ここで、

$$|g|_0^2 = (g, g) = (g, v - P_1 v) = (\nabla\psi, \nabla(v - P_1 v)) = (\nabla(\psi - P_1\psi), \nabla(v - P_1 v)) \leq |\psi - P_1\psi|_1 |v - P_1 v|_1.$$

Assumption 3.1 と (3.6) より

$$|g|_0^2 \leq C_0 h |\psi|_2 |v|_1$$

となる。次の Lemma 3.1 を用いると、

$$|g|_0 \leq C_0 h |v|_1$$

を得る。□

Lemma 3.1
Ω が区分的に滑らかな有界凸領域であり、

$$\begin{cases} -\triangle v = g & \text{in } \Omega, \\ v = 0 & \text{on } \partial\Omega \end{cases}$$

の解 v が $v \in H^2(\Omega)$ を満たすとする。このとき、

$$|v|_2 \leq |g|_0$$

が成り立つ。

Ω が 2 次元領域のときの証明は中尾、山本 [16] の p.99 補題 5.2 に載っている。

3.2 事後評価

$[u, p]$, $[u_h, p_h]$ をそれぞれ (2.5), (3.1) の解とする。また、

$$e_h = u - u_h, \quad \varepsilon_h = p - p_h$$

とする。まず $\delta(e_h, \varepsilon_h)$ の上界を評価する。

任意の $[v, q] \in \mathscr{S}$ に対して、

$$\mathscr{L}([e_h, \varepsilon_h], [v, q]) = \nu(\nabla e_h, \nabla v) - (\varepsilon_h, \operatorname{div} v) - (q, \operatorname{div} e_h) \tag{3.7}$$

である。また、(2.5) で $[v,q]=[\xi_h,q_h]$ とした式から (3.1) で $[v_h,q_h]=[\xi_h,q_h]$ とした式を引くと、任意の $[\xi_h,q_h]\in S_h\times Y_h$ に対して、

$$\nu(\nabla e_h,\nabla\xi_h)-(\varepsilon_h,\operatorname{div}\xi_h)-(q_h,\operatorname{div}e_h)=0 \tag{3.8}$$

を得る。(3.8) で $q_h=0$ ととると、任意の $\xi\in S_h$ に対して、

$$\nu(\nabla e_h,\nabla\xi_h)-(\varepsilon,\operatorname{div}\xi_h)=0. \tag{3.9}$$

(3.7) から (3.9) を引くと、

$$\begin{aligned}\mathscr{L}([e_h,\varepsilon_h],[v,q]) &= \nu(\nabla e_h,\nabla(v-\xi_h))-(\varepsilon_h,\operatorname{div}(v-\xi_h))-(q,\operatorname{div}e_h)\\ &= \nu(\nabla(u-u_h),\nabla(v-\xi_h))-(p-p_h,\operatorname{div}(v-\xi_h))+(q,\operatorname{div}u_h)\\ &= \nu(\overline{\nabla}u_h-\nabla u_h),\nabla(v-\xi_h))+\nu(\nabla u-\overline{\nabla}u_h,\nabla(v-\xi_h))\\ &- (p-p_h,\operatorname{div}(v-\xi_h))+(q,\operatorname{div}u_h).\end{aligned}$$

ここで、(2.5) より

$$\nu(\nabla u,\nabla v)-(p,\operatorname{div}v)=(f,v),\quad \forall v\in H_0^1(\Omega)^2$$

となることを用いると、

$$\begin{aligned}\mathscr{L}([e_h,\varepsilon_h],[v,q]) &= \nu(\overline{\nabla}u_h-\nabla u_h,\nabla(v-\xi_h))+(f+\nu\overline{\triangle}u_h-\nabla p_h,v-\xi_h)+(q,\operatorname{div}u_h)\\ &\le \nu|\overline{\nabla}u_h-\nabla u_h|_0|v-\xi_h|_1+|\nu\overline{\triangle}u_h-\nabla p_h+f|_0|v-\xi_h|_0+|\operatorname{div}u_h|_0|q|_0.\end{aligned}$$

ここで、$\xi_h\in S_h$ を $v=(v_1,v_2)$ の H_0^1-射影であるとする。すなわち、$\xi_h=(P_1v_1,P_1v_2)^T$ とする。すると、(3.5), (3.6) より、

$$\begin{aligned}\mathscr{L}([e_h,\varepsilon_h],[v,q]) &\le \nu|\overline{\nabla}u_h-\nabla u_h|_0|v|_1+|\nu\overline{\triangle}u_h-\nabla p_h+f|_0C_0h|v|_1+|\operatorname{div}u_h|_0|q|_0\\ &\le (\nu|\overline{\nabla}u_h-\nabla u_h|_0+C_0h|\nu\overline{\triangle}u_h-\nabla p_h+f|_0+|\operatorname{div}u_h|_0)(|v|_1+|q|_0).\end{aligned}$$

このようにして次の結果を得る。

Lemma 3.2
$[u,p]$, $[u_h,p_h]$ をそれぞれ (2.5), (3.1) の解として $e_h=u-u_h$, $\varepsilon_h=p-p_h$ とする。さらに C_0 を Assumption 3.1 の定数とする。このとき任意の 0 でない $[v,q]\in\mathscr{S}$ に対して次式が成り立つ。

$$\frac{\mathscr{L}([e_h,\varepsilon_h],[v,q])}{|v|_1+|q|_0}\le\nu|\overline{\nabla}u_h-\nabla u_h|_0+C_0h|\nu\overline{\triangle}u_h-\nabla p_h+f|_0+|\operatorname{div}u_h|_0.$$

Theorem 2.4 と Lemma 3.2 から次に述べる Stokes 方程式の有限要素解の事後誤差限界を得る。

Theorem 3.1 (事後誤差限界)
$[u,p]$, $[u_h,p_h]$ をそれぞれ (2.5), (3.1) の解とする。また β は (2.13) を満たす定数とする。さらに C_0 を Assumption 3.1 の定数とする。このとき次の不等式が成り立つ。

$$\begin{aligned} |u-u_h|_1 &\le \left(\frac{1}{\nu^2}+\frac{1}{\beta^2}\right)^{\frac{1}{2}} C(u_h,p_h), \\ |p-p_h|_0 &\le \left(\frac{1}{\beta}+\frac{\nu}{\beta^2}\right) C(u_h,p_h). \end{aligned} \tag{3.10}$$

ここで、$C(u_h,p_h)$ は有限要素解を用いて次のように定められる量である。

$$C(u_h,p_h) := \nu|\overline{\nabla}u_h - \nabla u_h|_0 + C_0 h|\nu\overline{\triangle}u_h - \nabla p_h + f|_0 + |\operatorname{div} u_h|_0. \tag{3.11}$$

証明
Theorem 2.4 において u を $u-u_h=e_h$, p を $p-p_h=\varepsilon_h$ として用いると、

$$\begin{aligned} |u-u_h|_1 &\le \left(\frac{1}{\nu^2}+\frac{1}{\beta^2}\right)^{\frac{1}{2}} \delta(e_h,\varepsilon_h), \\ |p-p_h|_0 &\le \left(\frac{1}{\beta}+\frac{\nu}{\beta^2}\right) \delta(e_h,\varepsilon_h) \end{aligned}$$

を得る。また、Lemma 3.2 を用いると、

$$\delta(e_h,\varepsilon_h) := \sup_{\substack{[v,q]\in\mathscr{S},\\ [v,q]\neq 0}} \frac{\mathscr{L}([e_h,\varepsilon_h],[v,q])}{|v|_1+|q|_0} \le C(u_h,p_h)$$

を得る。□

4 構成的アプリオリ誤差評価

前節では Stokes 問題の有限要素解の事後誤差限界について述べた。本節では前節と同様の方法を用いて、2 種類の構成的アプリオリ誤差の限界の導出方法を考え、アプリオリ定数の評価の計算手順を述べる。

4.1 アプリオリ評価

最初の方法は、前節で述べた事後誤差限界の結果を元にした手法である。任意の $f=[f_1,f_2]\in L^2(\Omega)^2$ に対して、$P_0 f\in S_h$ を

$$P_0 f = [P_0 f_1, P_0 f_2]^T$$

によって定義する。L^2-射影の性質から、

$$|f-P_0 f|_0^2 = |f|_0^2 - |P_0 f|_0^2.$$

したがって、$0 \leq \theta \leq \dfrac{\pi}{2}$ を満たすある θ が存在して、

$$\begin{aligned} |P_0 f|_0 &= |f|_0 \sin\theta, \\ |f - P_0 f|_0 &= |f|_0 \cos\theta \end{aligned} \tag{4.1}$$

となる。ここで、$f \in L^2(\Omega)^2$ によらない定数 K_1,K_2,K_3 が存在して次を満たすと仮定する。

$$|\overline{\nabla} u_h - \nabla u_h|_0 \leq K_1 |P_0 f|_0, \tag{4.2}$$

$$|\nu \overline{\triangle} u_h - \nabla p_h + P_0 f|_0 \leq K_2 |P_0 f|_0, \tag{4.3}$$

$$|\operatorname{div} u_h|_0 \leq K_3 |P_0 f|_0. \tag{4.4}$$

これらの定数は、一般化固有値問題を数値計算を用いて解くことにより決定される。詳しくは次の小節以降で述べる。このとき次の定理を得る。

Theorem 4.1 (アプリオリ誤差限界 I)
$[u,p]$, $[u_h,p_h]$ をそれぞれ (2.5), (3.1) の解とする。また β は (2.13) をみたす定数とし、C_0 を Assumption 3.1 の定数とする。さらに K_1, K_2, K_3 を (4.2), (4.3), (4.4) をみたす定数とする。このとき任意の $f \in L^2(\Omega)^2$ に対して次式が成り立つ。

$$\begin{aligned} |u - u_h|_1 &\leq \left(\frac{1}{\nu^2} + \frac{1}{\beta^2} \right)^{\frac{1}{2}} C_1(h) |f|_0, \\ |p - p_h|_0 &\leq \left(\frac{1}{\beta} + \frac{\nu}{\beta^2} \right) C_1(h) |f|_0. \end{aligned} \tag{4.5}$$

ここで、

$$C_1(h) := \sqrt{(\nu K_1 + C_0 h K_2 + K_3)^2 + (C_0 h)^2} \tag{4.6}$$

である。

証明
任意の $f \in L^2(\Omega)^2$ に対して、(3.11) より

$$C(u_h, p_h) = \nu |\overline{\nabla} u_h - \nabla u_h|_0 + C_0 h |\nu \overline{\triangle} u_h - \nabla p_h + P_0 f + f - P_0 f|_0 + |\operatorname{div} u_h|_0.$$

ここで、(4.2), (4.3), (4.4) を用いると、

$$C(u_h, p_h) \leq \nu K_1 |P_0 f|_0 + C_0 h (K_2 |P_0 f|_0 + |f - P_0 f|_0) + K_3 |P_0 f|_0.$$

さらに (4.1) を用いると、

$$\begin{aligned} C(u_h, p_h) &\leq ((\nu K_1 + C_0 h K_2 + K_3) \sin\theta + C_0 h \cos\theta) |f|_0 \\ &\leq \sqrt{(\nu K_1 + C_0 h K_2 + K_3)^2 + (C_0 h)^2} |f|_0 \\ &= C_1(h) |f|_0 \end{aligned}$$

を得る。□

2 つ目の方法は、$\overline{\nabla} u_h$ と $\overline{\triangle} u_h$ を用いることなく直接に導く方法である。(3.7) から (3.9) を引くと、任意の $[v,q] \in \mathscr{S}$ と任意の $\xi_h \in S_h$ に対して、

$$\mathscr{L}([e_h, \varepsilon_h], [v,q]) = \nu(\nabla(u-u_h), \nabla(v-\xi_h)) - (p-p_h, \mathrm{div}(v-\xi_h)) + (q, \mathrm{div}\, u_h)$$

を得る。ここで、

$$\xi_h := v_h = [P_1 v_1, P_1 v_2]^T, \quad (v = [v_1, v_2]^T)$$

ととる。すると H_0^1-射影の性質より、

$$\mathscr{L}([e_h, \varepsilon_h], [v,q]) = \nu(\nabla u, \nabla(v-v_h)) - (p-p_h, \mathrm{div}(v-v_h)) + (q, \mathrm{div}\, u_h).$$

ここで、(2.5) より

$$\nu(\nabla u, \nabla v) - (p, \mathrm{div}\, v) = (f, v), \quad \forall v \in H_0^1(\Omega)^2$$

となることを用いると、

$$\begin{aligned} \mathscr{L}([e_h, \varepsilon_h], [v,q]) &= (f - \nabla p_h, v - v_h) + (q, \mathrm{div}\, u_h) \\ &\leq |f - \nabla p_h|_0 |v - v_h|_0 + |q|_0 |\,\mathrm{div}\, u_h|_0. \end{aligned}$$

さらに (3.6) を用いると、

$$\begin{aligned} \mathscr{L}([e_h, \varepsilon_h], [v,q]) &\leq |f - \nabla p_h|_0 C_0 h |v|_1 + |q|_0 |\,\mathrm{div}\, u_h|_0 \\ &\leq (C_0 h |f - \nabla p_h|_0 + |\,\mathrm{div}\, u_h|_0)(|v|_1 + |q|_0). \end{aligned}$$

ゆえに次の Lemma が成り立つ。

Lemma 4.1
$[u,p]$, $[u_h, p_h]$ をそれぞれ (2.5), (3.1) の解として $e_h = u - u_h$, $\varepsilon_h = p - p_h$ とする。さらに C_0 を Assumption 3.1 の定数とする。このとき 0 でない任意の $[v,q] \in \mathscr{S}$ に対して次式が成り立つ。

$$\frac{\mathscr{L}([e_h, \varepsilon_h], [v,q])}{|v|_1 + |q|_0} \leq C_0 h |f - \nabla p_h|_0 + |\,\mathrm{div}\, u_h|_0.$$

ゆえに、Lemma 4.1 の結果から、Theorem 3.1 の $C(u_h, p_h)$ を以下のようにとることもできる。

$$C(u_h, p_h) = C_0 h |f - \nabla p_h|_0 + |\,\mathrm{div}\, u_h|_0.$$

ここで、$f \in L^2(\Omega)^2$ によらない定数 K_4 が存在して次を満たすと仮定する。

$$|-\nabla p_h + P_0 f|_0 \leq K_4 |P_0 f|_0. \tag{4.7}$$

K_4 を決定するための方法は後で述べる。これからもう 1 つのアプリオリ誤差評価式を得る。

Theorem 4.2 (アプリオリ誤差限界 II)
$[u,p]$, $[u_h,p_h]$ をそれぞれ (2.5), (3.1) の解とする。また β は (2.13) をみたす定数とし、C_0 を Assumption 3.1 の定数とする。さらに K_3, K_4 を (4.4), (4.7) をみたす定数とする。このとき任意の $f \in L^2(\Omega)^2$ に対して次式が成り立つ。

$$\begin{aligned} |u-u_h|_1 &\le \left(\frac{1}{\nu^2}+\frac{1}{\beta^2}\right)^{\frac{1}{2}} C_2(h)|f|_0, \\ |p-p_h|_0 &\le \left(\frac{1}{\beta}+\frac{\nu}{\beta^2}\right) C_2(h)|f|_0. \end{aligned} \tag{4.8}$$

ここで、

$$C_2(h) := \sqrt{(C_0hK_4+K_3)^2+(C_0h)^2} \tag{4.9}$$

である。

証明
任意の $f \in L^2(\Omega)^2$ に対して、

$$\begin{aligned} C(u_h,p_h) &= C_0h|f-\nabla p_h|_0 + |\operatorname{div} u_h|_0 \\ &= C_0h|-\nabla p_h + P_0f + f - P_0f|_0 + |\operatorname{div} u_h|_0 \\ &\le C_0h|-\nabla p_h + P_0f| + C_0h|f-P_0f|_0 + |\operatorname{div} u_h|_0. \end{aligned}$$

ここで、(4.4), (4.7) を用いると、

$$C(u_h,p_h) \le C_0h(K_4|P_0f|_0 + |f-P_0f|_0) + K_3|P_0f|.$$

さらに (4.1) を用いると、

$$\begin{aligned} C(u_h,p_h) &\le ((C_0hK_4+K_3)\sin\theta + C_0h\cos\theta)|f|_0 \\ &\le \sqrt{(C_0hK_4+K_3)^2+(C_0h)^2}|f|_0 \\ &= C(h)|f|_0 \end{aligned}$$

を得る。□

4.2 定数 $C_1(h)$, $C_2(h)$ の計算

この節では (4.6), (4.9) の定数 $C_1(h)$, $C_2(h)$ に現れる K_1, K_2, K_3, K_4 の具体的な計算方法を述べる。

$\dim X_h = n$, $\dim X_h^* = \hat{n}$, $\dim Y_h = m$ とする。$X_h \subsetneq X_h^*$ であるから $\hat{n} > n$ である。$\{\phi_j\}_{1\le j\le n}$ を X_h の基底関数、$\{\hat{\phi}_j\}_{1\le j\le \hat{n}}$ を X_h^* の基底関数、$\{\psi_j\}_{1\le j\le m}$ を Y_h の基底関数とする。このとき、実係数 $\{a_j^{(1)}\}_{1\le j\le n}$, $\{a_j^{(2)}\}_{1\le j\le n}$, $\{b_j\}_{1\le j\le m}$ を用いて有限要素解 $u_h = [u_h^{(1)}, u_h^{(2)}]^T \in S_h$, $p_h \in Y_h$ は次の形に一意に表される。

$$u_h^{(1)} = \sum_{i=1}^{n} a_i^{(1)}\phi_i,\ u_h^{(2)} = \sum_{i=1}^{n} a_i^{(2)}\phi_i,\ p_h = \sum_{i=1}^{m} b_i\psi_i.$$

このとき、X_h, Y_h の基底関数を用いて、(3.1) を書き直す。(3.1) より

$$\begin{aligned}
\nu(\nabla u_h^{(1)}, \nabla v_h^{(1)}) - \left(p_h, \frac{\partial v_h^{(1)}}{\partial x}\right) &= (f_1, v_h^{(1)}), \qquad \forall v_h^{(1)} \in X_h, \\
\nu(\nabla u_h^{(2)}, \nabla v_h^{(2)}) - \left(p_h, \frac{\partial v_h^{(2)}}{\partial y}\right) &= (f_2, v_h^{(2)}), \qquad \forall v_h^{(2)} \in X_h, \\
-\left(q_h, \frac{\partial u_h^{(1)}}{\partial x}\right) - \left(q_h, \frac{\partial u_h^{(2)}}{\partial y}\right) &= 0, \qquad \forall q_h \in Y_h
\end{aligned}$$

となるので、

$$\begin{aligned}
\nu \sum_{i=1}^{n} a_i^{(1)} (\nabla\phi_i, \nabla\phi_j) - \sum_{i=1}^{m} b_i \left(\psi_i, \frac{\partial \phi_j}{\partial x}\right) &= (f_1, \phi_j), \qquad 1 \le j \le n, \\
\nu \sum_{i=1}^{n} a_i^{(2)} (\nabla\phi_i, \nabla\phi_j) - \sum_{i=1}^{m} b_i \left(\psi_i, \frac{\partial \phi_j}{\partial y}\right) &= (f_2, \phi_j), \qquad 1 \le j \le n, \\
-\sum_{i=1}^{n} a_i^{(1)} \left(\psi_j, \frac{\partial \phi_i}{\partial x}\right) - \sum_{i=1}^{n} a_i^{(2)} \left(\psi_j, \frac{\partial \phi_i}{\partial y}\right) &= 0, \qquad 1 \le j \le m
\end{aligned} \tag{4.10}$$

を得る。以降 $m \times n$ 型の実行列を $\mathbf{R}^{m\times n}$ により表す。

$$\begin{aligned}
\boldsymbol{a}_1 &= (a_1^{(1)}, a_2^{(1)}, \ldots, a_n^{(1)}) \in \mathbf{R}^{1\times n}, \\
\boldsymbol{a}_2 &= (a_1^{(2)}, a_2^{(2)}, \ldots, a_n^{(2)}) \in \mathbf{R}^{1\times n}, \\
\boldsymbol{a} &= (\boldsymbol{a}_1, \boldsymbol{a}_2) \in \mathbf{R}^{1\times 2n}, \\
\boldsymbol{b} &= (b_1, b_2, \ldots, b_m) \in \mathbf{R}^{1\times m}, \\
\boldsymbol{f}_1 &= ((f_1, \phi_1), (f_1, \phi_2), \ldots, (f_1, \phi_n))^T \in \mathbf{R}^{n\times 1}, \\
\boldsymbol{f}_2 &= ((f_2, \phi_1), (f_2, \phi_2), \ldots, (f_2, \phi_n))^T \in \mathbf{R}^{n\times 1}, \\
\boldsymbol{f} &= \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{pmatrix} \in \mathbf{R}^{2n\times 1}
\end{aligned}$$

とする。さらに、

$$\begin{aligned}
D_0 &= ((\nabla\phi_i, \nabla\phi_j)) \in \mathbf{R}^{n\times n}, \\
D &= \begin{pmatrix} \nu D_0 & O \\ O & \nu D_0 \end{pmatrix} \in \mathbf{R}^{2n\times 2n}, \\
E_x &= \left(\left(\psi_i, \frac{\partial \phi_j}{\partial x}\right)\right) = -\left(\left(\frac{\partial \psi_i}{\partial x}, \phi_j\right)\right) \in \mathbf{R}^{m\times n}, \\
E_y &= \left(\left(\psi_i, \frac{\partial \phi_j}{\partial y}\right)\right) = -\left(\left(\frac{\partial \psi_i}{\partial y}, \phi_j\right)\right) \in \mathbf{R}^{m\times n}, \\
E &= (E_x\ E_y) \in \mathbf{R}^{m\times 2n}, \\
G &= \begin{pmatrix} D & -E^T \\ -E & O \end{pmatrix} \in \mathbf{R}^{(2n+m)\times(2n+m)}
\end{aligned}$$

とする。すると、(4.10) は

$$\begin{pmatrix} \nu D_0 & O & -(E_x)^T \\ O & \nu D_0 & -(E_y)^T \\ -E_x & -E_y & O \end{pmatrix} \begin{pmatrix} \boldsymbol{a}_1^T \\ \boldsymbol{a}_2^T \\ \boldsymbol{b}^T \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f_2} \\ \boldsymbol{0} \end{pmatrix}$$

となる。よって (3.1) は次の方程式と同値となる。

$$G \begin{pmatrix} \boldsymbol{a}^T \\ \boldsymbol{b}^T \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{pmatrix}. \tag{4.11}$$

対称行列 G は可逆であると仮定する。このことは、たいてい離散 inf-sup 条件から示すことができるが、数値計算で確認することもできる。G^{-1} を次のように表す。

$$G^{-1} = \begin{pmatrix} G_a & G_b^T \\ G_b & G_* \end{pmatrix} \in \mathbf{R}^{(2n+m)\times(2n+m)}.$$

ここで、G_a は $2n \times 2n$ 行列、G_b は $m \times 2n$ 行列、G_* は $m \times m$ 行列である。このとき (3.1) を満たす有限要素解 $[u_h, p_h] \in S_h \times Y_h$ の係数は、

$$\boldsymbol{a}^T = G_a \boldsymbol{f}, \quad \boldsymbol{b}^T = G_b \boldsymbol{f} \tag{4.12}$$

と表される。

次に u_h の係数を用いて $\overline{\nabla} u_h$ を表す。$\overline{\nabla} u_h = (\overline{\nabla} u_h^{(1)}, \overline{\nabla} u_h^{(2)}) \in (X_h^*)^2 \times (X_h^*)^2$ であるので、実係数 $\left\{c_j^{(1)}\right\}_{1\le j\le \hat{n}}$, $\left\{c_j^{(2)}\right\}_{1\le j\le \hat{n}}$, $\left\{d_j^{(1)}\right\}_{1\le j\le \hat{n}}$, $\left\{d_j^{(2)}\right\}_{1\le j\le \hat{n}}$ を用いて $\overline{\nabla} u_h$ は一意的に次のように書ける。

$$\begin{aligned} \overline{\nabla} u_h^{(1)} &= \left(\sum_{i=1}^{\hat{n}} c_i^{(1)} \hat{\phi}_i, \sum_{i=1}^{\hat{n}} d_i^{(1)} \hat{\phi}_i \right)^T, \\ \overline{\nabla} u_h^{(2)} &= \left(\sum_{i=1}^{\hat{n}} c_i^{(2)} \hat{\phi}_i, \sum_{i=1}^{\hat{n}} d_i^{(2)} \hat{\phi}_i \right)^T. \end{aligned}$$

また、

$$\begin{aligned} \boldsymbol{c}_1 &= (c_1^{(1)}, c_2^{(1)}, \ldots, c_{\hat{n}}^{(1)}) \in \mathbf{R}^{1\times\hat{n}}, \\ \boldsymbol{c}_2 &= (c_1^{(2)}, c_2^{(2)}, \ldots, c_{\hat{n}}^{(2)}) \in \mathbf{R}^{1\times\hat{n}}, \\ \boldsymbol{d}_1 &= (d_1^{(1)}, d_2^{(1)}, \ldots, d_{\hat{n}}^{(1)}) \in \mathbf{R}^{1\times\hat{n}}, \\ \boldsymbol{d}_2 &= (d_1^{(2)}, d_2^{(2)}, \ldots, d_{\hat{n}}^{(2)}) \in \mathbf{R}^{1\times\hat{n}} \end{aligned}$$

とする。L^2-射影の定義より

$$\begin{aligned} \sum_{i=1}^{\hat{n}} c_i^{(1)} (\hat{\phi}_i, \hat{\phi}_j) &= \sum_{i=1}^{n} a_i^{(1)} \left(\frac{\partial \phi_i}{\partial x}, \hat{\phi}_j \right), \quad 1 \le j \le \hat{n}, \\ \sum_{i=1}^{\hat{n}} d_i^{(1)} (\hat{\phi}_i, \hat{\phi}_j) &= \sum_{i=1}^{n} a_i^{(1)} \left(\frac{\partial \phi_i}{\partial y}, \hat{\phi}_j \right), \quad 1 \le j \le \hat{n} \end{aligned}$$

となる。なぜなら第 1 式目については、$\sum_{i=1}^{\hat{n}} c_i^{(1)}\hat{\phi}_i = \hat{P}_0\dfrac{\partial u_h^{(1)}}{\partial x}$ より、$\hat{P}_0$ の定義から

$$\left(\sum_{i=1}^{\hat{n}} c_i^{(1)}\hat{\phi}_i, \hat{\phi}_j\right) = \left(\hat{P}_0\frac{\partial u_h^{(1)}}{\partial x}, \hat{\phi}_j\right) = \left(\frac{\partial u_h^{(1)}}{\partial x}, \hat{\phi}_j\right) = \left(\sum_{i=1}^{n} a_i^{(1)}\frac{\partial \phi_i}{\partial x}, \hat{\phi}_j\right), \quad 1 \le j \le \hat{n}$$

となるからである。第 2 式目についても同様である。よって、

$$\begin{aligned}
\hat{L} &= ((\hat{\phi}_i, \hat{\phi}_j)) \in \mathbf{R}^{\hat{n}\times\hat{n}}, \\
K^x &= \left(\left(\frac{\partial \phi_i}{\partial x}, \hat{\phi}_j\right)\right) = -\left(\left(\phi_i, \frac{\partial \hat{\phi}_j}{\partial x}\right)\right) \in \mathbf{R}^{n\times\hat{n}}, \\
K^y &= \left(\left(\frac{\partial \phi_i}{\partial y}, \hat{\phi}_j\right)\right) = -\left(\left(\phi_i, \frac{\partial \hat{\phi}_j}{\partial y}\right)\right) \in \mathbf{R}^{n\times\hat{n}}
\end{aligned}$$

とおくと、

$$\boldsymbol{c}_1\hat{L} = \boldsymbol{a}_1 K^x, \quad \boldsymbol{d}_1\hat{L} = \boldsymbol{a}_1 K^y$$

を得る。また同様にして、

$$\boldsymbol{c}_2\hat{L} = \boldsymbol{a}_2 K^x, \quad \boldsymbol{d}_2\hat{L} = \boldsymbol{a}_2 K^y$$

を得る。ゆえに $\hat{L}$ の可逆性から次の関係が成り立つ。

Lemma 4.2
$n \times \hat{n}$ 行列 $M^x,\ M^y$ を $M^x = K^x\hat{L}^{-1}$, $M^y = K^y\hat{L}^{-1}$ によって定義する。このとき、$\overline{\nabla} u_h$ の係数は以下によって表される。

$$\begin{aligned}
\boldsymbol{c}_1 &= \boldsymbol{a}_1 M^x, \quad \boldsymbol{d}_1 = \boldsymbol{a}_1 M^y, \\
\boldsymbol{c}_2 &= \boldsymbol{a}_2 M^x, \quad \boldsymbol{d}_2 = \boldsymbol{a}_2 M^y.
\end{aligned} \tag{4.13}$$

次に、$\boldsymbol{f}$ を用いて $|P_0 f|_0^2$, $|\overline{\nabla} u_h - \nabla u_h|_0$, $|\nu\overline{\triangle} u_h - \nabla p_h + P_0 f|_0$, $|\operatorname{div} u_h|_0$, $|-\nabla p_h + P_0 f|_0$ の評価方法を述べる。

Lemma 4.3
$n \times n$ 行列 L と $2n \times 2n$ 行列 F を

$$\begin{aligned}
L &= ((\phi_i, \phi_j)) \in \mathbf{R}^{n\times n}, \\
F &= \begin{pmatrix} L^{-1} & O \\ O & L^{-1} \end{pmatrix} \in \mathbf{R}^{2n\times 2n}
\end{aligned}$$

によって定義する。このとき $|P_0 f|_0^2$ は次式によって表される。

$$|P_0 f|_0^2 = \boldsymbol{f}^T F \boldsymbol{f}. \tag{4.14}$$

証明
$f=[f_1,f_2]^T$ に対して、実係数 $\left\{q_j^{(1)}\right\}_{1\le j\le n},\ \left\{q_j^{(2)}\right\}_{1\le j\le n}$ を用いて

$$P_0f_1=\sum_{i=1}^n q_i^{(1)}\phi_i,\quad P_0f_2=\sum_{i=1}^n q_i^{(2)}\phi_i$$

と表す。ここで、

$$\boldsymbol{q}_1=(q_1^{(1)},\dots,q_n^{(1)})\in\mathbf{R}^{1\times n},\quad \boldsymbol{q}_2=(q_1^{(2)},\dots,q_n^{(2)})\in\mathbf{R}^{1\times n}$$

とする。すると、

$$\begin{aligned}L\boldsymbol{q}_1^T &= \left(\sum_{i=1}^n(\phi_1,\phi_i)q_i^{(1)},\cdots,\sum_{i=1}^n(\phi_n,\phi_i)q_i^{(1)}\right)^T\\ &= ((P_0f_1,\phi_1),\dots,(P_0f_1,\phi_n))^T\\ &= ((f_1,\phi_1),\dots,(f_1,\phi_n))^T\\ &= \boldsymbol{f}_1\end{aligned}$$

となる。よって、

$$\boldsymbol{q}_1^T=L^{-1}\boldsymbol{f}_1.$$

同様に、

$$\boldsymbol{q}_2^T=L^{-1}\boldsymbol{f}_2.$$

ゆえに、

$$\begin{aligned}|P_0f|_0^2 &= (P_0f_1,P_0f_1)+(P_0f_2,P_0f_2)\\ &= \sum_{i=1}^n q_i^{(1)}\sum_{j=1}^n L_{ij}q_j^{(1)}+\sum_{i=1}^n q_i^{(2)}\sum_{j=1}^n L_{ij}q_j^{(2)}\\ &= \boldsymbol{q}_1L\boldsymbol{q}_1^T+\boldsymbol{q}_2L\boldsymbol{q}_2^T\\ &= \boldsymbol{f}_1^TL^{-1}\boldsymbol{f}_1+\boldsymbol{f}_2L^{-1}\boldsymbol{f}_2\\ &= \boldsymbol{f}^TF\boldsymbol{f}.\qquad\square\end{aligned}$$

Lemma 4.4
$2n\times 2n$ 行列 $Q_1,\ A_1$ を

$$\begin{aligned}Q_1 &= \begin{pmatrix}D_0-M^x(K^x)^T-M^y(K^y)^T & O\\ O & D_0-M^x(K^x)^T-M^y(K^y)^T\end{pmatrix}\in\mathbf{R}^{2n\times 2n},\\ A_1 &= (G_aQ_1G_a)\in\mathbf{R}^{2n\times 2n}\end{aligned}$$

によって定義する。このとき

$$K_1=\left(\sup_{\boldsymbol{x}\in\mathbf{R}^{2n}}\frac{\boldsymbol{x}^TA_1\boldsymbol{x}}{\boldsymbol{x}^TF\boldsymbol{x}}\right)^{\frac{1}{2}}\tag{4.15}$$

とおくと、(4.2) の不等式が成り立つ。

証明

$$
\begin{aligned}
|\overline{\nabla} u_h - \nabla u_h|_0^2 &= (\overline{\nabla} u_h - \nabla u_h, \overline{\nabla} u_h - \nabla u_h) \\
&= (\overline{\nabla} u_h - \nabla u_h, \overline{\nabla} u_h) - (\overline{\nabla} u_h - \nabla u_h, \nabla u_h).
\end{aligned}
$$

ここで、射影の性質から上式の右辺第一項は 0 となる。よって、

$$
\begin{aligned}
|\overline{\nabla} u_h - \nabla u_h|_0^2 &= (\nabla u_h, \nabla u_h) - (\overline{\nabla} u_h, \nabla u_h) \\
&= (\nabla u_h, \nabla u_h) - (\overline{\nabla} u_h, \overline{\nabla} u_h) \\
&= (\nabla u_h^{(1)}, \nabla u_h^{(1)}) + (\nabla u_h^{(2)}, \nabla u_h^{(2)}) - \{(\overline{\nabla} u_h^{(1)}, \overline{\nabla} u_h^{(1)}) + (\overline{\nabla} u_h^{(2)}, \overline{\nabla} u_h^{(2)})\} \\
&= \sum_{i=1}^{n} a_i^{(1)} \sum_{j=1}^{n} (\nabla \phi_i, \nabla \phi_j) a_j^{(1)} + \sum_{i=1}^{n} a_i^{(2)} \sum_{j=1}^{n} (\nabla \phi_i, \nabla \phi_j) a_j^{(2)} - \left\{ \sum_{i=1}^{\hat{n}} c_i^{(1)} \sum_{j=1}^{\hat{n}} (\hat{\phi}_i, \hat{\phi}_j) c_j^{(1)} \right. \\
&+ \left. \sum_{i=1}^{\hat{n}} d_i^{(1)} \sum_{j=1}^{\hat{n}} (\hat{\phi}_i, \hat{\phi}_j) d_j^{(1)} + \sum_{i=1}^{\hat{n}} c_i^{(2)} \sum_{j=1}^{\hat{n}} (\hat{\phi}_i, \hat{\phi}_j) c_j^{(2)} + \sum_{i=1}^{\hat{n}} d_i^{(2)} \sum_{j=1}^{\hat{n}} (\hat{\phi}_i, \hat{\phi}_j) d_j^{(2)} \right\} \\
&= \boldsymbol{a}_1 D_0 \boldsymbol{a}_1^T + \boldsymbol{a}_2 D_0 \boldsymbol{a}_2^T - \boldsymbol{c}_1 \hat{L} \boldsymbol{c}_1^T - \boldsymbol{d}_1 \hat{L} \boldsymbol{d}_1^T - \boldsymbol{c}_2 \hat{L} \boldsymbol{c}_2^T - \boldsymbol{d}_2 \hat{L} \boldsymbol{d}_2^T.
\end{aligned}
$$

さらに、Lemma 4.2 を用いると、

$$
\begin{aligned}
|\overline{\nabla} u_h - \nabla u_h|_0^2 &= \boldsymbol{a}_1 D_0 \boldsymbol{a}_1^T + \boldsymbol{a}_2 D_0 \boldsymbol{a}_2^T \\
&- \boldsymbol{a_1} (M^x (K^x)^T + M^y (K^y)^T) \boldsymbol{a_1}^T - \boldsymbol{a_2} (M^x (K^x)^T + M^y (K^y)^T) \boldsymbol{a_2}^T \\
&= (\boldsymbol{a}_1 \ \boldsymbol{a_2}) \begin{pmatrix} D_0 - M^x (K^x)^T - M^y (K^y)^T & O \\ O & D_0 - M^x (K^x)^T - M^y (K^y)^T \end{pmatrix} \begin{pmatrix} \boldsymbol{a}_1^T \\ \boldsymbol{a}_2^T \end{pmatrix} \\
&= \boldsymbol{a} Q_1 \boldsymbol{a}^T \\
&= \boldsymbol{f}^T G_a Q_1 G_a \boldsymbol{f} \\
&= \boldsymbol{f}^T A_1 \boldsymbol{f}.
\end{aligned}
$$

よって、

$$
\frac{|\overline{\nabla} u_h - \nabla u_h|_0}{|P_0 f|_0} = \left(\frac{\boldsymbol{f}^T A_1 \boldsymbol{f}}{\boldsymbol{f}^T F \boldsymbol{f}} \right)^{\frac{1}{2}}
$$

となるので K_1 を

$$
K_1 \geq \left(\sup_{\boldsymbol{x} \in \mathbf{R}^{2n}} \frac{\boldsymbol{x}^T A_1 \boldsymbol{x}}{\boldsymbol{x}^T F \boldsymbol{x}} \right)^{\frac{1}{2}}
$$

を満たすように取れば (4.2) の不等式が成り立つ。□

次に、

$$
\hat{D}^{xx} = \left(\left(\frac{\partial \hat{\phi}_i}{\partial x}, \frac{\partial \hat{\phi}_j}{\partial x} \right) \right) \in \mathbf{R}^{\hat{n} \times \hat{n}}, \ \hat{D}^{xy} = \left(\left(\frac{\partial \hat{\phi}_i}{\partial x}, \frac{\partial \hat{\phi}_j}{\partial y} \right) \right) \in \mathbf{R}^{\hat{n} \times \hat{n}}, \ \hat{D}^{yy} = \left(\left(\frac{\partial \hat{\phi}_i}{\partial y}, \frac{\partial \hat{\phi}_j}{\partial y} \right) \right) \in \mathbf{R}^{\hat{n} \times \hat{n}}
$$

と定義し、

$$
\hat{E}^{xx} = M^x \hat{D}^{xx} (M^x)^T \in \mathbf{R}^{n \times n}, \quad \hat{E}^{xy} = M^x \hat{D}^{xy} (M^y)^T \in \mathbf{R}^{n \times n}, \quad \hat{E}^{yy} = M^y \hat{D}^{yy} (M^y)^T \in \mathbf{R}^{n \times n}
$$

とする。さらに、

$$E_1 = \begin{pmatrix} \hat{E}^{xx} + \hat{E}^{xy} + (\hat{E}^{xy})^T + \hat{E}^{yy} & O \\ O & \hat{E}^{xx} + \hat{E}^{xy} + (\hat{E}^{xy})^T + \hat{E}^{yy} \end{pmatrix} \in \mathbf{R}^{2n\times 2n}$$

とする。また、

$$\tilde{D} = ((\nabla\psi_i, \nabla\psi_j)) \in \mathbf{R}^{m\times m},$$

$$\hat{F}^{xx} = \left(\left(\frac{\partial\hat{\phi}_i}{\partial x}, \frac{\partial\psi_j}{\partial x}\right)\right) \in \mathbf{R}^{\hat{n}\times m}, \qquad \hat{F}^{xy} = \left(\left(\frac{\partial\hat{\phi}_i}{\partial x}, \frac{\partial\psi_j}{\partial y}\right)\right) \in \mathbf{R}^{\hat{n}\times m},$$
$$\hat{F}^{yx} = \left(\left(\frac{\partial\hat{\phi}_i}{\partial y}, \frac{\partial\psi_j}{\partial x}\right)\right) \in \mathbf{R}^{\hat{n}\times m}, \qquad \hat{F}^{yy} = \left(\left(\frac{\partial\hat{\phi}_i}{\partial y}, \frac{\partial\psi_j}{\partial y}\right)\right) \in \mathbf{R}^{\hat{n}\times m}$$

と定義し、

$$E_2 = \begin{pmatrix} (M^x\hat{F}^{xx} + M^y\hat{F}^{yx})G_b \\ (M^x\hat{F}^{xy} + M^y\hat{F}^{yy})G_b \end{pmatrix} \in \mathbf{R}^{2n\times 2n},$$
$$E_3 = \begin{pmatrix} -(K^x\hat{L}^{-1}(K^x)^T + K^y\hat{L}^{-1}(K^y)^T)L^{-1} & O \\ O & -(K^x\hat{L}^{-1}(K^x)^T + K^y\hat{L}^{-1}(K^y)^T)L^{-1} \end{pmatrix} \in \mathbf{R}^{2n\times 2n}$$

とする。

Lemma 4.5
$2n \times 2n$ 行列 A_2 を

$$\begin{aligned} A_2 &= \nu^2(G_a)^T E_1 G_a - \nu G_a E_2 - \nu(G_a E_2)^T + \nu G_a E_3 + \nu(G_a E_3)^T \\ &+ G_b^T EF + (G_b^T EF)^T + G_b^T \tilde{D} G_b + F \end{aligned}$$

によって定義する。このとき

$$K_2 = \left(\sup_{\boldsymbol{x}\in\mathbf{R}^{2n}} \frac{\boldsymbol{x}^T A_2 \boldsymbol{x}}{\boldsymbol{x}^T F \boldsymbol{x}}\right)^{\frac{1}{2}} \tag{4.16}$$

とおくと、(4.3) の不等式が成り立つ。

証明
$|\nu\overline{\triangle}u_h - \nabla p_h + P_0 f|_0^2$ を展開すると、

$$\begin{aligned} |\nu\overline{\triangle}u_h - \nabla p_h + P_0 f|_0^2 &= (\nu\overline{\triangle}u_h - \nabla p_h + P_0 f, \nu\overline{\triangle}u_h - \nabla p_h + P_0 f) \\ &= \nu^2(\overline{\triangle}u_h, \overline{\triangle}u_h) - \nu(\overline{\triangle}u_h, \nabla p_h) - \nu(\nabla p_h, \overline{\triangle}u_h) \\ &+ \nu(\overline{\triangle}u_h, P_0 f) + \nu(P_0 f, \overline{\triangle}u_h) - (\nabla p_h, P_0 f) \\ &- (P_0 f, \nabla p_h) + (\nabla p_h, \nabla p_h) + |P_0 f|_0^2 \end{aligned}$$

となる。ここで、上式の各項を $\boldsymbol{f}$ の 2 次形式により表す。

$$
\begin{aligned}
(\overline{\triangle} u_h, \overline{\triangle} u_h) &= (\operatorname{div} \overline{\nabla} u_h, \operatorname{div} \overline{\nabla} u_h) \\
&= (\operatorname{div} \overline{\nabla} u_h^{(1)}, \operatorname{div} \overline{\nabla} u_h^{(1)}) + (\operatorname{div} \overline{\nabla} u_h^{(2)}, \operatorname{div} \overline{\nabla} u_h^{(2)}) \\
&= \sum_{i=1}^{\hat{n}} c_i^{(1)} \sum_{j=1}^{\hat{n}} \left(\frac{\partial \hat{\phi}_i}{\partial x}, \frac{\partial \hat{\phi}_j}{\partial x} \right) c_j^{(1)} + \sum_{i=1}^{\hat{n}} c_i^{(1)} \sum_{j=1}^{\hat{n}} \left(\frac{\partial \hat{\phi}_i}{\partial x}, \frac{\partial \hat{\phi}_j}{\partial y} \right) d_j^{(1)} \\
&+ \sum_{i=1}^{\hat{n}} d_i^{(1)} \sum_{j=1}^{\hat{n}} \left(\frac{\partial \hat{\phi}_i}{\partial y}, \frac{\partial \hat{\phi}_j}{\partial x} \right) c_j^{(1)} + \sum_{i=1}^{\hat{n}} d_i^{(1)} \sum_{j=1}^{\hat{n}} \left(\frac{\partial \hat{\phi}_i}{\partial y}, \frac{\partial \hat{\phi}_j}{\partial y} \right) d_j^{(1)} \\
&+ \sum_{i=1}^{\hat{n}} c_i^{(2)} \sum_{j=1}^{\hat{n}} \left(\frac{\partial \hat{\phi}_i}{\partial x}, \frac{\partial \hat{\phi}_j}{\partial x} \right) c_j^{(2)} + \sum_{i=1}^{\hat{n}} c_i^{(2)} \sum_{j=1}^{\hat{n}} \left(\frac{\partial \hat{\phi}_i}{\partial x}, \frac{\partial \hat{\phi}_j}{\partial y} \right) d_j^{(2)} \\
&+ \sum_{i=1}^{\hat{n}} d_i^{(2)} \sum_{j=1}^{\hat{n}} \left(\frac{\partial \hat{\phi}_i}{\partial y}, \frac{\partial \hat{\phi}_j}{\partial x} \right) c_j^{(2)} + \sum_{i=1}^{\hat{n}} d_i^{(2)} \sum_{j=1}^{\hat{n}} \left(\frac{\partial \hat{\phi}_i}{\partial y}, \frac{\partial \hat{\phi}_j}{\partial y} \right) d_j^{(2)} \\
&= \boldsymbol{c}_1 \hat{D}^{xx} \boldsymbol{c}_1^T + \boldsymbol{c}_1 \hat{D}^{xy} \boldsymbol{d}_1^T + \boldsymbol{d}_1 (\hat{D}^{xy})^T \boldsymbol{c}_1^T + \boldsymbol{d}_1 \hat{D}^{yy} \boldsymbol{d}_1^T \\
&+ \boldsymbol{c}_2 \hat{D}^{xx} \boldsymbol{c}_2^T + \boldsymbol{c}_2 \hat{D}^{xy} \boldsymbol{d}_2^T + \boldsymbol{d}_2 (\hat{D}^{xy})^T \boldsymbol{c}_2^T + \boldsymbol{d}_2 \hat{D}^{yy} \boldsymbol{d}_2^T \\
&= \boldsymbol{a}_1 \hat{E}^{xx} \boldsymbol{a}_1^T + \boldsymbol{a}_1 \hat{E}^{xy} \boldsymbol{a}_1^T + \boldsymbol{a}_1 (\hat{E}^{xy})^T \boldsymbol{a}_1^T + \boldsymbol{a}_1 \hat{E}^{yy} \boldsymbol{a}_1^T \\
&+ \boldsymbol{a}_2 \hat{E}^{xx} \boldsymbol{a}_2^T + \boldsymbol{a}_2 \hat{E}^{xy} \boldsymbol{a}_2^T + \boldsymbol{a}_2 (\hat{E}^{xy})^T \boldsymbol{a}_2^T + \boldsymbol{a}_2 \hat{E}^{yy} \boldsymbol{a}_2^T \\
&= \boldsymbol{a} E_1 \boldsymbol{a}^T \\
&= \boldsymbol{f}^T (G_a)^T E_1 G_a \boldsymbol{f}.
\end{aligned}
$$

$$
\begin{aligned}
(\overline{\triangle} u_h, \nabla p_h) &= (\operatorname{div} \overline{\nabla} u_h, \nabla p_h) \\
&= \left(\operatorname{div} \overline{\nabla} u_h^{(1)}, \frac{\partial p_h}{\partial x} \right) + \left(\operatorname{div} \overline{\nabla} u_h^{(2)}, \frac{\partial p_h}{\partial y} \right) \\
&= \sum_{i=1}^{\hat{n}} c_i^{(1)} \sum_{j=1}^{m} \left(\frac{\partial \hat{\phi}_i}{\partial x}, \frac{\partial \psi_j}{\partial x} \right) b_j + \sum_{i=1}^{\hat{n}} d_i^{(1)} \sum_{j=1}^{m} \left(\frac{\partial \hat{\phi}_i}{\partial y}, \frac{\partial \psi_j}{\partial x} \right) b_j \\
&+ \sum_{i=1}^{\hat{n}} c_i^{(2)} \sum_{j=1}^{m} \left(\frac{\partial \hat{\phi}_i}{\partial x}, \frac{\partial \psi_j}{\partial y} \right) b_j + \sum_{i=1}^{\hat{n}} d_i^{(2)} \sum_{j=1}^{m} \left(\frac{\partial \hat{\phi}_i}{\partial y}, \frac{\partial \psi_j}{\partial y} \right) b_j \\
&= \boldsymbol{c}_1 \hat{F}^{xx} \boldsymbol{b}^T + \boldsymbol{d}_1 \hat{F}^{yx} \boldsymbol{b}^T + \boldsymbol{c}_2 \hat{F}^{xy} \boldsymbol{b}^T + \boldsymbol{d}_2 \hat{F}^{yy} \boldsymbol{b}^T \\
&= \boldsymbol{a}_1 (M^x \hat{F}^{xx} + M^y \hat{F}^{yx}) \boldsymbol{b}^T + \boldsymbol{a}_2 (M^x \hat{F}^{xy} + M^y \hat{F}^{yy}) \boldsymbol{b}^T \\
&= (\boldsymbol{a}_1 \ \boldsymbol{a}_2) \begin{pmatrix} (M^x \hat{F}^{xx} + M^y \hat{F}^{yx}) G_b \\ (M^x \hat{F}^{xy} + M^y \hat{F}^{yy}) G_b \end{pmatrix} \boldsymbol{f} \\
&= \boldsymbol{a} E_2 \boldsymbol{f} \\
&= \boldsymbol{f}^T G_a E_2 \boldsymbol{f}.
\end{aligned}
$$

$$
(\nabla p_h, \overline{\triangle} u_h) = (\boldsymbol{f}^T G_a E_2 \boldsymbol{f})^T = \boldsymbol{f}^T (G_a E_2)^T \boldsymbol{f}.
$$

$$
\begin{aligned}
& (\overline{\triangle} u_h, P_0 f) \\
= \; & (\operatorname{div} \overline{\nabla} u_h^{(1)}, P_0 f_1) + (\operatorname{div} \overline{\nabla} u_h^{(2)}, P_0 f_2) \\
= \; & \sum_{i=1}^{\hat{n}} c_i^{(1)} \sum_{j=1}^{n} \left(\frac{\partial \hat{\phi}_i}{\partial x}, \phi_j \right) q_j^{(1)} + \sum_{i=1}^{\hat{n}} d_i^{(1)} \sum_{j=1}^{n} \left(\frac{\partial \hat{\phi}_i}{\partial y}, \phi_j \right) q_j^{(1)} \\
+ \; & \sum_{i=1}^{\hat{n}} c_i^{(2)} \sum_{j=1}^{n} \left(\frac{\partial \hat{\phi}_i}{\partial x}, \phi_j \right) q_j^{(2)} + \sum_{i=1}^{\hat{n}} d_i^{(2)} \sum_{j=1}^{n} \left(\frac{\partial \hat{\phi}_i}{\partial y}, \phi_j \right) q_j^{(2)} \\
= \; & \boldsymbol{c}_1 (-K^x)^T \boldsymbol{q}_1^T + \boldsymbol{d}_1 (-K^y)^T \boldsymbol{q}_1^T + \boldsymbol{c}_2 (-K^x)^T \boldsymbol{q}_2^T + \boldsymbol{d}_2 (-K^y)^T \boldsymbol{q}_2^T \\
= \; & (\boldsymbol{a}_1 \; \boldsymbol{a}_2) \begin{pmatrix} -(M^x (K^x)^T + M^y (K^y)^T) L^{-1} & O \\ O & -(M^x (K^x)^T + M^y (K^y)^T) L^{-1} \end{pmatrix} \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{pmatrix} \\
= \; & \boldsymbol{a} \begin{pmatrix} -(K^x \hat{L}^{-1} (K^x)^T + K^y \hat{L}^{-1} (K^y)^T) L^{-1} & O \\ O & -(K^x \hat{L}^{-1} (K^x)^T + K^y \hat{L}^{-1} (K^y)^T) L^{-1} \end{pmatrix} \boldsymbol{f} \\
= \; & \boldsymbol{f}^T G_a E_3 \boldsymbol{f}.
\end{aligned}
$$

$$
(P_0 f, \overline{\triangle} u_h) = (\boldsymbol{f}^T G_a E_3 \boldsymbol{f})^T = \boldsymbol{f}^T (G_a E_3)^T \boldsymbol{f}.
$$

$$
\begin{aligned}
(\nabla p_h, P_0 f) &= \left(\frac{\partial p_h}{\partial x}, P_0 f_1 \right) + \left(\frac{\partial p_h}{\partial y}, P_0 f_2 \right) \\
&= \sum_{i=1}^{m} b_i \sum_{j=1}^{n} \left(\frac{\partial \psi_i}{\partial x}, \phi_j \right) q_j^{(1)} + \sum_{i=1}^{m} b_i \sum_{j=1}^{n} \left(\frac{\partial \psi_i}{\partial y}, \phi_j \right) q_j^{(2)} \\
&= \boldsymbol{b} (-E_x) \boldsymbol{q}_1^T + \boldsymbol{b} (-E_y) \boldsymbol{q}_2^T \\
&= \boldsymbol{b} (-E_x L^{-1} \boldsymbol{f}_1 - E_y L^{-1} \boldsymbol{f}_2) \\
&= -\boldsymbol{b} (E_x \; E_y) \begin{pmatrix} L^{-1} & O \\ O & L^{-1} \end{pmatrix} \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{pmatrix} \\
&= -\boldsymbol{f}^T G_b^T E F \boldsymbol{f}.
\end{aligned}
$$

$$
(P_0 f, \nabla p_h) = (-\boldsymbol{f}^T G_b^T E F \boldsymbol{f})^T = -\boldsymbol{f} (G_b^T E F)^T \boldsymbol{f}.
$$

$$
\begin{aligned}
(\nabla p_h, \nabla p_h) &= \left(\frac{\partial p_h}{\partial x}, \frac{\partial p_h}{\partial x} \right) + \left(\frac{\partial p_h}{\partial y}, \frac{\partial p_h}{\partial y} \right) \\
&= \sum_{i=1}^{m} b_i \sum_{j=1}^{m} \left(\frac{\partial \psi_i}{\partial x}, \frac{\partial \psi_j}{\partial x} \right) b_j + \sum_{i=1}^{m} b_i \sum_{j=1}^{m} \left(\frac{\partial \psi_i}{\partial y}, \frac{\partial \psi_j}{\partial y} \right) b_j \\
&= \boldsymbol{b} \tilde{D} \boldsymbol{b}^T \\
&= \boldsymbol{f}^T G_b^T \tilde{D} G_b \boldsymbol{f}.
\end{aligned}
$$

ゆえに、

$$
|\nu \overline{\triangle} u_h - \nabla p_h + P_0 f|_0^2 = \boldsymbol{f}^T A_2 \boldsymbol{f}.
$$

よって、

$$
\frac{|\nu \overline{\triangle} u_h - \nabla p_h + P_0 f|_0}{|P_0 f|_0} = \left(\frac{\boldsymbol{f}^T A_2 \boldsymbol{f}}{\boldsymbol{f}^T F \boldsymbol{f}} \right)^{\frac{1}{2}}
$$

となるので K_2 を

$$K_2 \geq \left(\sup_{\boldsymbol{x} \in \mathbf{R}^{2n}} \frac{\boldsymbol{x}^T A_2 \boldsymbol{x}}{\boldsymbol{x}^T F \boldsymbol{x}} \right)^{\frac{1}{2}}$$

を満たすように取れば (4.3) の不等式が成り立つ。□

次に、

$$D^{xx} = \left(\left(\frac{\partial \phi_i}{\partial x}, \frac{\partial \phi_j}{\partial x} \right) \right) \in \mathbf{R}^{n \times n},\ D^{xy} = \left(\left(\frac{\partial \phi_i}{\partial x}, \frac{\partial \phi_j}{\partial y} \right) \right) \in \mathbf{R}^{n \times n},\ D^{yy} = \left(\left(\frac{\partial \phi_i}{\partial y}, \frac{\partial \phi_j}{\partial y} \right) \right) \in \mathbf{R}^{n \times n}$$

とする。また、

$$Q_3 = \begin{pmatrix} D^{xx} & D^{xy} \\ (D^{xy})^T & D^{yy} \end{pmatrix} \in \mathbf{R}^{2n \times 2n}$$

とする。このとき、これらの行列を用いて次の Lemma を得る。

Lemma 4.6
$2n \times 2n$ 行列 A_3 を

$$A_3 = (G_a Q_3 G_a) \in \mathbf{R}^{2n \times 2n}$$

によって定義する。このとき

$$K_3 = \left(\sup_{\boldsymbol{x} \in \mathbf{R}^{2n}} \frac{\boldsymbol{x}^T A_3 \boldsymbol{x}}{\boldsymbol{x}^T F \boldsymbol{x}} \right)^{\frac{1}{2}} \tag{4.17}$$

とおくと (4.4) の不等式が成り立つ。

証明

$$\begin{aligned}
|\operatorname{div} u_h|_0^2 &= (\operatorname{div} u_h, \operatorname{div} u_h) \\
&= \left(\frac{\partial u_h^{(1)}}{\partial x}, \frac{\partial u_h^{(1)}}{\partial x} \right) + \left(\frac{\partial u_h^{(1)}}{\partial x}, \frac{\partial u_h^{(2)}}{\partial y} \right) + \left(\frac{\partial u_h^{(2)}}{\partial y}, \frac{\partial u_h^{(1)}}{\partial x} \right) + \left(\frac{\partial u_h^{(2)}}{\partial y}, \frac{\partial u_h^{(2)}}{\partial y} \right) \\
&= \boldsymbol{a}_1 D^{xx} \boldsymbol{a}_1^T + \boldsymbol{a}_1 D^{xy} \boldsymbol{a}_2^T + \boldsymbol{a}_2 (D^{xy})^T \boldsymbol{a}_1^T + \boldsymbol{a}_2 D^{yy} \boldsymbol{a}_2^T \\
&= \boldsymbol{a} Q_3 \boldsymbol{a}^T \\
&= \boldsymbol{f}^T G_a Q_3 G_a \boldsymbol{f} \\
&= \boldsymbol{f}^T A_3 \boldsymbol{f}.
\end{aligned}$$

よって、

$$\frac{|\operatorname{div} u_h|_0}{|P_0 f|_0} = \left(\frac{\boldsymbol{f}^T A_3 \boldsymbol{f}}{\boldsymbol{f}^T F \boldsymbol{f}} \right)^{\frac{1}{2}}$$

となるので K_3 を

$$K_3 \geq \left(\sup_{\boldsymbol{x} \in \mathbf{R}^{2n}} \frac{\boldsymbol{x}^T A_3 \boldsymbol{x}}{\boldsymbol{x}^T F \boldsymbol{x}} \right)^{\frac{1}{2}}$$

を満たすように取れば (4.4) の不等式が成り立つ。□

Lemma 4.7
$2n \times 2n$ 行列 A_4 を

$$A_4 = (G_b^T EF + (G_b^T EF)^T + G_b^T \tilde{D} G_b + F) \in \mathbf{R}^{2n \times 2n}$$

によって定義する。このとき

$$K_4 = \left(\sup_{\boldsymbol{x} \in \mathbf{R}^{2n}} \frac{\boldsymbol{x}^T A_4 \boldsymbol{x}}{\boldsymbol{x}^T F \boldsymbol{x}} \right)^{\frac{1}{2}} \tag{4.18}$$

とおくと (4.7) の不等式が成り立つ。

証明

$$| - \nabla p_h + P_0 f|_0^2 \;=\; (\nabla p_h, \nabla p_h) - (\nabla p_h, P_0 f) - (P_0 f, \nabla p_h) + |P_0 f|_0^2.$$

ここで、Lemma 4.5 の証明内の結果より、

$$\begin{aligned} | - \nabla p_h + P_0 f|_0^2 &= \boldsymbol{f}^T G_b^T EF \boldsymbol{f} + \boldsymbol{f}^T (G_b^T EF)^T \boldsymbol{f} + \boldsymbol{f}^T G_b^T \tilde{D} G_b \boldsymbol{f} + \boldsymbol{f}^T F \boldsymbol{f} \\ &= \boldsymbol{f}^T (G_b^T EF + (G_b EF)^T + G_b^T \tilde{D} G_b + F) \boldsymbol{f} \\ &= \boldsymbol{f}^T A_4 \boldsymbol{f}. \end{aligned}$$

よって、

$$\frac{| - \nabla p_h + P_0 f|_0}{|P_0 f|_0} = \left(\frac{\boldsymbol{f}^T A_4 \boldsymbol{f}}{\boldsymbol{f}^T F \boldsymbol{f}} \right)^{\frac{1}{2}}$$

となるので K_4 を

$$K_4 \geq \left(\sup_{\boldsymbol{x} \in \mathbf{R}^{2n}} \frac{\boldsymbol{x}^T A_4 \boldsymbol{x}}{\boldsymbol{x}^T F \boldsymbol{x}} \right)^{\frac{1}{2}}$$

を満たすように取れば (4.7) の不等式が成り立つ。□

(4.15)-(4.18) の評価は A を $2n \times 2n$ の実対称行列、B を $2n \times 2n$ の実対称正定値行列としたときの一般化固有値問題 $A\boldsymbol{x} = \lambda B \boldsymbol{x}$ の固有値の絶対値の最大値を求める問題に帰着できる。

5 Stokes 版 Aubin-Nitsche's trick

この節の主な内容は中尾、山本、渡部 [4] によるものである。ここでは、事後誤差評価と事前誤差評価式の結果に Aubin-Nitsche's trick に似た手法を適用することにより速度に関する L^2 誤差評価 $|u - u_h|_0$ を導く。

$[u, p]$ を (2.5) の解、$[u_h, p_h]$ を (3.1) の解とする。(1.1) において f を $u - u_h$ に置き換えた Stokes 方程式

$$\left\{ \begin{array}{rcll} -\nu \triangle \phi + \nabla \psi &=& u - u_h & \text{in } \Omega, \\ \operatorname{div} \phi &=& 0 & \text{in } \Omega, \\ \phi &=& 0 & \text{on } \partial\Omega \end{array} \right. \tag{5.1}$$

を考える。(2.5) と同様に

$$\nu(\nabla\phi, \nabla v) - (\psi, \operatorname{div} v) = (u - u_h, v), \quad \forall v \in H_0^1(\Omega)^2$$

を得る。ここで、$v = u - u_h$ とおくと、

$$(u - u_h, u - u_h) = \nu(\nabla(u - u_h), \nabla\phi) - (\operatorname{div} u_h, \psi) \tag{5.2}$$

を得る。一方、任意の $v_h \in X_h^2$ と $q_h \in Y_h$ に対して、$[u, p]$ は (2.5) の解であるので、

$$\nu(\nabla u, \nabla v_h) - (p, \operatorname{div} v_h) = (f, v_h). \tag{5.3}$$

また $[u_h, p_h]$ は (3.1) の解であるので、

$$\nu(\nabla u_h, \nabla v_h) - (p_h, \operatorname{div} v_h) - (q_h, \operatorname{div} u_h) = (f, v_h). \tag{5.4}$$

よって (5.3), (5.4) より

$$\nu(\nabla(u - u_h), \nabla v_h) + (q_h, \operatorname{div} u_h) - (p - p_h, \operatorname{div} v_h) = 0 \tag{5.5}$$

を得る。従って (5.2) と (5.5) より

$$\begin{aligned} (u - u_h, u - u_h) &= \nu(\nabla(u - u_h), \nabla(\phi - v_h)) + (\psi - q_h, \operatorname{div} u_h) + (p - p_h, \operatorname{div} v_h) \\ &\leq \nu|u - u_h|_1|\phi - v_h|_1 + |\operatorname{div} u_h|_0|\psi - q_h|_0 + |p - p_h|_0|\operatorname{div} v_h|_0 \end{aligned} \tag{5.6}$$

が導かれる。

ここで $v_h \in X_h^2$, $q_h \in Y_h$ を特に (5.1) の有限要素解として選ぶ。このとき Theorem 4.2 より

$$|\phi - v_h|_1 \leq \left(\frac{1}{\nu^2} + \frac{1}{\beta^2}\right)^{\frac{1}{2}} C_2(h)|u - u_h|_0, \tag{5.7}$$

$$|\psi - q_h|_0 \leq \left(\frac{1}{\beta} + \frac{\nu}{\beta^2}\right) C_2(h)|u - u_h|_0 \tag{5.8}$$

を得る。また、(4.4) と $|P_0 f|_0 \leq |f|_0$ より

$$|\operatorname{div} v_h|_0 \leq K_3|u - u_h|_0 \tag{5.9}$$

が成り立つ。よって、

$$C_2^{(1)} = \left(\frac{1}{\nu^2} + \frac{1}{\beta^2}\right)^{\frac{1}{2}} C_2(h), \tag{5.10}$$

$$C_2^{(2)} = \left(\frac{1}{\beta} + \frac{\nu}{\beta^2}\right) C_2(h) \tag{5.11}$$

と定めることにより (5.6)-(5.9) から $u - u_h$ に対する L^2 での事後誤差評価式が以下のように得られる。

Theorem 5.1
$C_2^{(1)}$, $C_2^{(2)}$ を (5.10), (5.11) で定める。また、K_3 を (4.4) をみたす定数とする。このとき (2.5) の解 $[u,p]$ と (3.1) の解 $[u_h,p_h]$ に対し次の事後誤差評価式が成り立つ。

$$|u-u_h|_0 \le \nu C_2^{(1)}|u-u_h|_1 + C_2^{(2)}|\operatorname{div} u_h|_0 + K_3|p-p_h|_0. \tag{5.12}$$

また Theorem 4.2 と (4.4) から

$$\begin{aligned} |u-u_h|_1 &\le C_2^{(1)}|f|_0, \\ |\operatorname{div} u_h|_0 &\le K_3|P_0 f|_0 \le K_3|f|_0, \\ |p-p_h|_0 &\le C_2^{(2)}|f|_0 \end{aligned}$$

を得る。(5.12) の右辺に上式を用いると以下の L^2 での事前誤差評価式を得る。

Theorem 5.2
$C_2^{(1)}$, $C_2^{(2)}$ を (5.10), (5.11) で定める。また、K_3 を (4.4) をみたす定数とする。このとき任意の $f \in L^2(\Omega)$ と (2.5) の解 $[u,p]$ と (3.1) の解 $[u_h,p_h]$ に対し次の事前誤差評価式が成り立つ。

$$|u-u_h|_0 \le (\nu C_2^{(1)^2} + 2C_2^{(2)}K_3)|f|_0. \tag{5.13}$$

6 一般化固有値問題

この節では、A を $n\times n$ の実対称行列、B を $n\times n$ の実対称正定値行列としたときの一般化固有値問題

$$A\boldsymbol{x} = \lambda B\boldsymbol{x}, \qquad (\lambda \in \mathbf{R},\ \mathbf{0} \neq \boldsymbol{x} \in \mathbf{R}^n) \tag{6.1}$$

の固有値の絶対値の最大値の上界を精度保証付きで評価するいくつかの手法を述べる。

6.1 一般化固有値問題の性質のまとめ

この節の主な内容は桂田 [15] を参考にした。この小節では後で用いる一般化固有値問題の性質をまとめておく。

B が実対称正定値行列である場合、Cholesky 分解

$$B = U^T U$$

を代入して (U は正則な上三角行列であり、特に対角成分が正であるものを選ぶことができ、その場合は一意的に決定される)

$$A\boldsymbol{x} = U^T U\boldsymbol{x}.$$

両辺に $(U^T)^{-1}$ をかけると

$$(U^T)^{-1}A\boldsymbol{x} = U\boldsymbol{x}.$$

ここで次の Lemma が成り立つ。

Lemma 6.1
P が正則であるとき、$(P^T)^{-1} = (P^{-1})^T$. 特に正則な対称行列の逆行列は対称行列である。

証明
$Q := P^{-1}$ とすると、$PQ = I$. これから $Q^T P^T = (PQ)^T = I^T = I$. ゆえに $(P^T)^{-1} = Q^T = (P^{-1})^T$. □

なお、$(P^T)^{-1}$ のことを P^{-T} と書く。

$$A' := (U^{-1})^T A U^{-1}, \quad \boldsymbol{y} := U\boldsymbol{x}$$

とおくと

$$\begin{aligned} A\boldsymbol{x} = \lambda B\boldsymbol{x} &\iff A'\boldsymbol{y} = \lambda U\boldsymbol{x}, \\ \boldsymbol{x} = 0 &\iff \boldsymbol{y} = 0 \end{aligned}$$

が成り立つ。A が実対称であれば、A' も実対称である。これは次の Lemma からわかる。

Lemma 6.2
P が n 次対称行列であるとき、任意の n 次正方行列 Q について $P' := Q^T P Q$ は対称である。

証明
$(P')^T = (Q^T P Q)^T = Q^T P^T (Q^T)^T = Q^T P Q = P'$. □

Lemma 6.3
A は実対称行列で、B は実対称正定値行列、$B = U^T U$ を Cholesky 分解 (すなわち U は上三角行列) とするとき、$A' := (U^{-1})^T A U^{-1}$ とおくと、

$$\det(\lambda I - A') = \det(\lambda I - B^{-1}A) \quad (\lambda \in \mathbf{C}).$$

証明

$$\begin{aligned} \det(\lambda I - A') &= \det(\lambda U U^{-1} - (U^{-1})^T A U^{-1}) \\ &= \det(\lambda U^{-1} U - U^{-1}(U^T)^{-1} A) \\ &= \det(\lambda I - (U^T U)^{-1} A) \\ &= \det(\lambda I - B^{-1} A). \ \square \end{aligned}$$

ここまでをまとめると、以下を得る。

Theorem 6.1
A が n 次実対称行列で、B が n 次実対称正定値行列であるとき B の Cholesky 分解 $B = U^TU$(U は対角成分が正である上三角行列) を用いて、

$$A' := (U^{-1})^T A U^{-1}$$

とおくと次の (1)-(3) が成り立つ。

(1) A' は実対称行列である (ゆえにその固有値はすべて実数であり、対応する固有ベクトルで R^n の正規直交基底をなすものが存在する)。

(2) A' の固有値は (A, B) の一般化固有値に等しい。すなわち、

$$\begin{aligned} \sigma(A') &:= \{\lambda \in \mathbf{C}\ ;\ \exists \boldsymbol{x} \in \mathbf{C}^n \setminus \{0\} \quad \text{s.t.} \quad A'\boldsymbol{x} = \lambda \boldsymbol{x}\}, \\ \sigma(A, B) &:= \{\mu \in \mathbf{C}\ ;\ \exists \boldsymbol{y} \in \mathbf{C}^n \setminus \{0\} \quad \text{s.t.} \quad A\boldsymbol{y} = \mu B\boldsymbol{y}\} \end{aligned}$$

とおくとき、$\sigma(A') = \sigma(A, B)$. さらに

$$\det(\lambda I - A') = \det(\lambda - B^{-1}A)$$

であるので重複度まで込めて等しい。

(3) A' の固有値を重複度の分だけ並べて書いたものを

$$\lambda_1, \lambda_2, \ldots, \lambda_n,$$

これらに対応する A' の固有ベクトルからなる $\mathbf{R}^n$ の正規直交基底を

$$\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$$

とするとき、

$$\boldsymbol{u}_i := U^{-1}\boldsymbol{v}_i \quad (i = 1, 2, \cdots, n)$$

とおくと、

$$\begin{aligned} A\boldsymbol{u}_i &= \lambda_i B\boldsymbol{u}_i \quad (i = 1, 2, \cdots, n), \\ \langle \boldsymbol{u}_i, \boldsymbol{u}_j \rangle &= \delta_{ij} \quad (i, j = 1, 2, \cdots, n). \end{aligned}$$

ただし $\langle \cdot, \cdot \rangle$ は次式で定義される $\mathbf{R}^n$ の内積である。

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle := (B\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}, B\boldsymbol{y}) = (U\boldsymbol{x}, U\boldsymbol{y}) \quad (\boldsymbol{x}, \boldsymbol{y} \in \mathbf{R}^n).$$

証明
(1), (2) と (3) の前半は済んでいる。(3) の後半については、

$$\langle \boldsymbol{u}_i, \boldsymbol{u}_j \rangle = (U\boldsymbol{u}_i, U\boldsymbol{u}_j) = (\boldsymbol{v}_i, \boldsymbol{v}_j) = \delta_{ij}. \ \square$$

実対称行列 A' の標準固有値問題については、Rayleigh 商

$$\frac{(A'\boldsymbol{y},\boldsymbol{y})}{(\boldsymbol{y},\boldsymbol{y})}=\frac{\boldsymbol{y}^TA'\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{y}}$$

が重要な役目を果たすが、これについては次の Lemma が成り立つ。

Lemma 6.4
A が n 次実対称行列で、B が n 次実対称正定値行列であるとき、B の Cholesky 分解 $B=U^TU$(U は対角成分が正である上三角行列) を用いて、

$$A':=(U^{-1})^TAU$$

とおくと、任意の $\boldsymbol{x}\in\mathbf{R}^n\setminus\{\mathbf{0}\}$ に対して

$$\frac{(A\boldsymbol{x},\boldsymbol{x})}{(B\boldsymbol{x},\boldsymbol{x})}=\frac{(A'\boldsymbol{y},\boldsymbol{y})}{(\boldsymbol{y},\boldsymbol{y})},\quad \boldsymbol{y}:=U\boldsymbol{x}.$$

証明
最初に $\boldsymbol{x}\neq 0\Leftrightarrow\boldsymbol{y}\neq 0$ であることを注意しておく。

$$\frac{(A\boldsymbol{x},\boldsymbol{x})}{(B\boldsymbol{x},\boldsymbol{x})}=\frac{(AU^{-1}\boldsymbol{y},U^{-1}\boldsymbol{y})}{(U\boldsymbol{x},\boldsymbol{Ux})}=\frac{((U^{-1})^TAU^{-1}\boldsymbol{y},\boldsymbol{y})}{(\boldsymbol{y},\boldsymbol{y})}=\frac{(A'\boldsymbol{y},\boldsymbol{y})}{(\boldsymbol{y},\boldsymbol{y})}.\square$$

Corollary 6.1
A が n 次実対称行列で、B が n 次実対称正定値行列であるとき、

$$\max\sigma(A,B)=\max_{\boldsymbol{x}\in\mathbf{R}^n\setminus\{0\}}\frac{(A\boldsymbol{x},\boldsymbol{x})}{(B\boldsymbol{x},\boldsymbol{x})},\quad \min\sigma(A,B)=\min_{\boldsymbol{x}\in\mathbf{R}^n\setminus\{0\}}\frac{(A\boldsymbol{x},\boldsymbol{x})}{(B\boldsymbol{x},\boldsymbol{x})},$$

$$\max\{|\lambda|\ ;\ \lambda\in\sigma(A,B)\}=\max_{\boldsymbol{x}\in\mathbf{R}^n\setminus\{0\}}\left|\frac{(A\boldsymbol{x},\boldsymbol{x})}{(B\boldsymbol{x},\boldsymbol{x})}\right|.$$

より詳しく min-max 原理も成り立つ。すなわち固有値全部を大きいほうから順に (重複度の分だけ繰り返して) 並べたものを

$$\lambda_1\geq\lambda_2\geq\cdots\geq\lambda_n$$

とするとき、$1\leq k\leq n$ なる k に対して、

$$\begin{aligned}\max_{\dim V=k}\min_{\boldsymbol{x}\in V\setminus\{0\}}\frac{(A\boldsymbol{x},\boldsymbol{x})}{(B\boldsymbol{x},\boldsymbol{x})} &= \text{大きい方から } k \text{ 番目の固有値} = \lambda_k,\\ \min_{\dim V=k}\max_{\boldsymbol{x}\in V\setminus\{0\}}\frac{(A\boldsymbol{x},\boldsymbol{x})}{(B\boldsymbol{x},\boldsymbol{x})} &= \text{小さい方から } k \text{ 番目の固有値} = \lambda_{n-k+1}.\end{aligned}$$

ここで、$\dim V=k$ は V が $\mathbf{R}^n$ のすべての k 次元部分空間を走ることを意味する。

6.2 アルゴリズム

この節の主な内容は渡部、山本、中尾 [10] によるものである。この小節では一般化固有値問題の固有値の絶対値の最大値の上界を精度保証付きで評価するアルゴリズムを述べる。

6.2.1 アルゴリズムに関する注釈

(6.1) の固有値の絶対値の最大値を評価する問題は、Corollary 6.1 より

$$\gamma := \sup_{0 \neq \boldsymbol{x} \in \mathbf{R}^n} \left| \frac{\boldsymbol{x}^T A \boldsymbol{x}}{\boldsymbol{x}^T B \boldsymbol{x}} \right| \tag{6.2}$$

を評価することと同値である。以下 γ にできるだけ近い上界を精度保証付きで求める方法を考える。

浮動小数点演算による数値計算の結果に含まれる丸め誤差の厳密な評価を得るために区間演算を用いる。区間演算では数値に含まれる誤差を考慮するため実数 x を下限 $\underline{x}$ と上界 $\overline{x}$ による区間

$$[\overline{x}, \underline{x}] = \{x \mid \underline{x} \leq x \leq \overline{x}\}$$

で表現し、すべての演算を区間に対する演算に置き換える。従って以降、行列 A, B は各成分が区間となる区間行列とし、A に含まれるすべての実対称行列および B に含まれるすべての実対称正定値行列に対して (6.2) を評価するアルゴリズムを考える。

アルゴリズム中の "$interval$" は区間演算による丸め誤差を考慮した厳密な計算を、"$floating$" は浮動小数点演算による近似計算を表す。ここでの「厳密」とは丸め誤差のない無限桁の計算を行った結果が区間演算によって真に包み込まれているという意味である。例えば「B を区間演算により CC^T と Cholesky 分解する」とは「B に含まれる実対称正定値行列を丸め誤差なく Cholesky 分解して得られる下三角行列をすべて含むような区間行列 C を求める」ことである。また区間 $x = [\underline{x}, \overline{x}]$ の絶対値を $|x| := \max\{|\overline{x}|, |\underline{x}|\}$ と定義する。正のスカラーとして計算する値 $(\lambda_1,\ \lambda_2,\ \lambda_1\lambda_2)$ は、区間演算の結果として決まる区間の上限を採用する。

区間行列に対して浮動小数点演算による近似計算を行う場合は、各要素の中点をとった行列を用いる。また "$\tilde{T}$" のようにチルダ付きのものは近似計算により得られた行列を、"C" のようにチルダが付いていない行列は各要素が区間となる行列を表す。スカラーについても同様とする。また、任意の行列 F に対して $r(F)$ で F のスペクトル半径を表す。

行列の 2-ノルムを

$$\|F\|_2 := \sup_{\boldsymbol{x}^T \boldsymbol{x} = 1} |F\boldsymbol{x}|,$$

(ベクトルノルムはユークリッドノルム), l_∞-ノルム (∞-ノルム) を

$$\|F\|_\infty := \max_{1 \leq i \leq n} \sum_{j=1}^{n} |F_{ij}|$$

と定義する。一般に $r(F) \leq \|F\|_\infty$ が成り立つ。

6.2.2 近似対角化法

Theorem 6.1 から、一般化固有値問題 (6.1) は実対称正定値行列 B の Cholesky 分解 $B = CC^T$ により

$$C^{-1}AC^{-T}\boldsymbol{x} = \lambda\boldsymbol{x}$$

に変形することができる。ここで、C は正則な下三角行列である。Lemma 6.1 の後に述べたとおり、$C^{-T} := (C^{-1})^T$ である。

近似対角化法 (approximate diagonalization method) は、一般化固有値問題を標準固有値問題に変形したのちに近似的な対角化を行い、行列ノルムの計算により固有値を評価する方法である。

アルゴリズム：ADM(近似対角化法)

Step 1. B を CC^T と Cholesky 分解 $(interval)$.

Step 2. $E := C^{-1}AC^{-T}$ を計算 $(interval)$.

Step 3. E の正規化された固有ベクトルにより、近似対角化行列 $\tilde{T}$ を作成 $(floating)$.

Step 4. $D := \tilde{T}^T E\tilde{T}, \quad \lambda_1 := \max\limits_{1\le i\le n}\sum\limits_{j=1}^{n}|D_{ij}|$ を計算 $(interval)$.

Step 5. $I := (\tilde{T}\tilde{T}^T)^{-1}, \quad \lambda_2 := \max\limits_{1\le i\le n}\sum\limits_{j=1}^{n}|I_{ij}|$ を計算 $(interval)$.

Step 6. $\lambda_1\lambda_2$ を計算 $(interval)$.

Theorem 6.2
アルゴリズム ADM で得られた $\lambda_1\lambda_2$ は (6.2) の γ の 1 つの上界を与える。

証明
A_0 を A に含まれる任意の実対称行列、B_0 を B に含まれる任意の実対称正定値行列とする。B_0 の正定値性と Step 1 より C に含まれる正則な下三角行列 C_0 が存在し、$B_0 = C_0C_0^T$ と書ける。$\boldsymbol{y} = C_0^T\boldsymbol{x}$ とおけば、

$$\sup_{0\neq\boldsymbol{x}\in\mathbf{R}^n}\left|\frac{\boldsymbol{x}^TA_0\boldsymbol{x}}{\boldsymbol{x}^TB_0\boldsymbol{x}}\right| = \sup_{0\neq\boldsymbol{y}\in\mathbf{R}^n}\left|\frac{(C_0^{-T}\boldsymbol{y})^TA_0C_0^{-T}\boldsymbol{y}}{(C_0^T\boldsymbol{x})^TC_0^T\boldsymbol{x}}\right| = \sup_{0\neq\boldsymbol{y}\in\mathbf{R}^n}\left|\frac{\boldsymbol{y}^TC_0^{-1}A_0C_0^{-T}\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{y}}\right|$$

を得る。ここで、$E_0 := C_0^{-1}A_0C_0^{-T}$ とおくと、A_0 が対称なので E_0 は E に含まれる実対称行列である。$\boldsymbol{z} := \tilde{T}^{-1}\boldsymbol{y}$ とおくと、

$$\sup_{0\neq\boldsymbol{y}\in\mathbf{R}^n}\left|\frac{\boldsymbol{y}^TE_0\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{y}}\right| = \sup_{0\neq\boldsymbol{z}\in\mathbf{R}^n}\left|\frac{(\tilde{T}\boldsymbol{z})^TE_0\tilde{T}\boldsymbol{z}}{(\tilde{T}\boldsymbol{z})^T\tilde{T}\boldsymbol{z}}\right| = \sup_{0\neq\boldsymbol{z}\in\mathbf{R}^n}\left|\frac{\boldsymbol{z}^T\tilde{T}^TE_0\tilde{T}\boldsymbol{z}}{\boldsymbol{z}^T\tilde{T}^T\tilde{T}\boldsymbol{z}}\right|$$

となる。$\boldsymbol{\xi} := \tilde{T}\boldsymbol{z}$ とおき正規化を行う。

$$\begin{aligned}
\sup_{0\neq\boldsymbol{z}\in\mathbf{R}^n}\left|\frac{\boldsymbol{z}^T\tilde{T}^TE_0\tilde{T}\boldsymbol{z}}{\boldsymbol{z}^T\tilde{T}^T\tilde{T}\boldsymbol{z}}\right| &= \sup_{0\neq\boldsymbol{z}\in\mathbf{R}^n}\left|\frac{\boldsymbol{z}^T\tilde{T}^TE_0\tilde{T}\boldsymbol{z}}{\boldsymbol{z}^T\boldsymbol{z}}\frac{\boldsymbol{z}^T\boldsymbol{z}}{\boldsymbol{z}^T\tilde{T}\tilde{T}\boldsymbol{z}}\right| \\
&\leq \sup_{\boldsymbol{z}^T\boldsymbol{z}=1}\left|\boldsymbol{z}^T\tilde{T}^TE_0\tilde{T}\boldsymbol{z}\right|\sup_{\boldsymbol{\xi}^T\boldsymbol{\xi}=1}\left|(\tilde{T}^{-1}\boldsymbol{\xi})^T\tilde{T}^{-1}\boldsymbol{\xi}\right| \\
&= \sup_{\boldsymbol{z}^T\boldsymbol{z}=1}\left|\boldsymbol{z}^T\tilde{T}^TE_0\tilde{T}\boldsymbol{z}\right|\sup_{\boldsymbol{\xi}^T\boldsymbol{\xi}=1}\left|\boldsymbol{\xi}^T(\tilde{T}\tilde{T}^T)^{-1}\boldsymbol{\xi}\right| \\
&= r\left(\tilde{T}^TE_0\tilde{T}\right)r\left((\tilde{T}\tilde{T}^T)^{-1}\right) \\
&\leq \|\tilde{T}^TE_0\tilde{T}\|_\infty\|(\tilde{T}\tilde{T}^T)^{-1}\|_\infty \\
&= \lambda_1\lambda_2.
\end{aligned}$$

上式より結論が導かれる。□

近似対角化法は、近似対角化行列の作用により標準固有値問題を丸め誤差がなければ対角行列である (従って対角行列が近いと期待される) 行列と、近似的に単位行列であることが期待される行列の固有値問題に帰着させる手法である。問題点として、行列の対角成分が非対角成分に比べて小さい場合に、Cholesky 分解に負数を含む区間の平方根が現れてしまうこと、Step 5 の I の計算において逆行列を厳密に評価する必要があり計算量が大きくなることがあげられる。

6.2.3 Rump の方法

Rump の方法も近似対角化法と同様、まず一般化固有値問題を標準固有値問題に変形する。次に固有値の絶対値の最大値を近似計算し、真の値を上回ることを期待して僅かに膨らませてそれが上界であることを判定する。以下 I_n を $n\times n$ の単位行列とする。

アルゴリズム：Rump(Rump の方法)

Step 1. B を CC^T と Cholesky 分解 $(interval)$.

Step 2. $E := C^{-1}AC^{-T}$ を計算 $(interval)$.

Step 3. E の固有値の絶対値の最大値 $\tilde{\beta}$ を近似計算 $(floating)$.

Step 4. ある微小な数 $0<\delta\ll 1$ により $\beta := (1+\delta)\tilde{\beta}$ とする $(interval)$.

Step 5. $X_1 := -E+\beta I_n,\quad X_2 := E+\beta I_n$ を計算 $(interval)$.

Step 6. X_h $(k=1,2)$ を Cholesky 分解する $(floating)$. Cholesky 分解が失敗する場合は δ の値を大きくして Step 4 に戻るか停止。

Step 7. $Y_h := \tilde{C}_k\tilde{C}_k^T - X_k,\quad \lambda_k = \max\limits_{1\leq i\leq n}\sum\limits_{j=1}^{n}|(Y_k)_{ij}|$ $(k=1,2)$ を計算 $(interval)$.

Step 8. $\beta+\max\{\lambda_1,\lambda_2\}$ を計算 $(interval)$.

Theorem 6.3
アルゴリズム Rump で得られた $\beta + \max\{\lambda_1, \lambda_2\}$ は (6.2) の γ の 1 つの上界を与える。

証明
A_0 を A に含まれる任意の実対称行列、B_0 を B に含まれる任意の実対称正定値行列とする。B_0 の正定値性と Step 1 より C に含まれる正則な下三角行列 C_0 が存在し、$B_0 = C_0 C_0^T$ と書ける。また Step 2 と A_0 の対称性より E に含まれる実対称行列 E_0 が存在する。ここで、$\boldsymbol{x}^T\boldsymbol{x} = 1$ となる任意の $\boldsymbol{x} \in \mathbf{R}^n$ に対して $\tilde{C}_1\tilde{C}_1^T + E_0 - \beta I_n$ が実対称行列であることより

$$|\boldsymbol{x}^T(\tilde{C}_1\tilde{C}_1^T + E_0 - \beta I_n)\boldsymbol{x}| \leq r\left(\tilde{C}_1\tilde{C}_1^T + E_0 - \beta I_n\right) \leq \|\tilde{C}_1\tilde{C}_1^T + E_0 - \beta I_n\|_\infty = \lambda_1$$

となる。同様に、

$$|\boldsymbol{x}^T(\tilde{C}_2\tilde{C}_2^T - E_0 - \beta I_n)\boldsymbol{x}| \leq \lambda_2$$

を得る。ここで、$\tilde{C}_i\tilde{C}_i^T$ $(i = 1, 2)$ は正定値である。なぜなら、まず形から対称性は明らか。また $\tilde{C}_i$ は Cholesky 分解の結果なので正則であり (対角成分がすべて正である下三角行列)、$\boldsymbol{x} \neq 0$ に対して $\tilde{C}_i^T\boldsymbol{x} \neq 0$ であるから

$$(\tilde{C}_i\tilde{C}_i^T\boldsymbol{x}, \boldsymbol{x}) = (\tilde{C}_i^T\boldsymbol{x}, \tilde{C}_i^T\boldsymbol{x}) > 0$$

となるので正定値である。$\tilde{C}_i\tilde{C}_i^T$ $(i = 1, 2)$ の正定値性と $\boldsymbol{x}^T\boldsymbol{x} = 1$ であることを用いると、

$$\lambda_1 \geq \boldsymbol{x}^T(\tilde{C}_1\tilde{C}_1^T + E_0 - \beta I_n)\boldsymbol{x} \geq \boldsymbol{x}^T E_0\boldsymbol{x} - \beta,$$
$$\lambda_2 \geq \boldsymbol{x}^T(\tilde{C}_2\tilde{C}_2^T - E_0 - \beta I_n)\boldsymbol{x} \geq -\boldsymbol{x}^T E_0\boldsymbol{x} - \beta.$$

従って、

$$-\beta - \lambda_2 \leq \boldsymbol{x}^T E_0\boldsymbol{x} \leq \beta + \lambda_1$$

となり結論が導かれる。□

Rump の方法は近似対角化法と同じく区間演算による Cholesky 分解を行うという問題点がある。ただしこれは標準固有値問題のアルゴリズムを一般化固有値問題に適用したからであり、標準固有値問題のスペクトル半径を評価するには優れた手法である。また、A が正定値の場合にはアルゴリズム中の X_2 に関する手順は省略可能である。

6.2.4 改良近似対角化法

改良近似対角化法 (approximate diagonalization method-advanced) は近似対角化法の修正版である。近似対角化行列を近似的に求めればよい点に着目し区間演算による Cholesky 分解を近似計算に置き換えることが特徴である。

アルゴリズム：ADM-a (改良近似対角化法)

Step 1. Cholesky 分解 $B \approx \tilde{C}\tilde{C}^T$ を行う $(floating)$.

Step 2. $\tilde{E} := \tilde{C}^{-1}A\tilde{C}^{-T}$ を計算 $(floating)$.

Step 3. $\tilde{E}$ の正規化された固有ベクトルにより近似対角化行列 $\tilde{T}$ を作成 $(floating)$.

Step 4. $\tilde{P} := \tilde{T}^T\tilde{C}^{-1}$ を計算 $(floating)$.

Step 5. $D := \tilde{P}A\tilde{P}^T, \quad \lambda_1 = \max\limits_{1\le i\le n}\sum\limits_{j=1}^{n}|D_{ij}|$ を計算 $(interval)$.

Step 6. $I := (\tilde{P}B\tilde{P}^T)^{-1}, \quad \lambda_2 = \max\limits_{1\le i\le n}\sum\limits_{j=1}^{n}|I_{ij}|$ を計算 $(interval)$.

Step 7. $\lambda_1\lambda_2$ を計算 $(interval)$.

Theorem 6.4
アルゴリズム ADM-a で得られた $\lambda_1\lambda_2$ は (6.2) の γ の 1 つの上界を与える。

証明
A_0 を A に含まれる任意の実対称行列、B_0 を B に含まれる任意の実対称正定値行列とする。$\boldsymbol{z} = \tilde{P}^{-T}\boldsymbol{x}$ とすれば

$$\begin{aligned}\sup_{0\neq\boldsymbol{x}\in\mathbf{R}^n}\left|\frac{\boldsymbol{x}^TA_0\boldsymbol{x}}{\boldsymbol{x}^TB_0\boldsymbol{x}}\right| &= \sup_{0\neq\boldsymbol{z}\in\mathbf{R}^n}\left|\frac{\boldsymbol{z}^T\tilde{P}A_0\tilde{P}^T\boldsymbol{z}}{\boldsymbol{z}^T\tilde{P}B_0\tilde{P}^T\boldsymbol{z}}\right| \\ &= \sup_{0\neq\boldsymbol{z}\in\mathbf{R}^n}\left|\frac{\boldsymbol{z}^T\tilde{P}A_0\tilde{P}^T\boldsymbol{z}}{\boldsymbol{z}^T\boldsymbol{z}}\frac{\boldsymbol{z}^T\boldsymbol{z}}{\boldsymbol{z}^T\tilde{P}B_0\tilde{P}^T\boldsymbol{z}}\right| \\ &\le \sup_{0\neq\boldsymbol{z}\in\mathbf{R}^n}\left|\frac{\boldsymbol{z}^T\tilde{P}A_0\tilde{P}^T\boldsymbol{z}}{\boldsymbol{z}^T\boldsymbol{z}}\right|\sup_{0\neq\boldsymbol{z}\in\mathbf{R}^n}\left|\frac{\boldsymbol{z}^T\boldsymbol{z}}{\boldsymbol{z}^T\tilde{P}B_0\tilde{P}^T\boldsymbol{z}}\right|.\end{aligned}$$

ここで、$I_0^T := \tilde{P}B_0\tilde{P}^T$ とおくと、I_0 は実対称正定値行列である。実際まず

$$I_0^T = (\tilde{P}B_0\tilde{P}^T)^T = (\tilde{P}^T)^TB_0\tilde{P}^T = \tilde{P}B_0\tilde{P}^T = I_0$$

であり、また $\boldsymbol{\xi} := \tilde{P}^T\boldsymbol{z}$ とおくと、B_0 の正定値性から

$$\boldsymbol{z}^TI_0\boldsymbol{z} = \boldsymbol{z}^T\tilde{P}B_0\tilde{P}^T\boldsymbol{z} = \left(\tilde{P}^T\boldsymbol{z}\right)^TB_0\left(\tilde{P}^T\boldsymbol{z}\right) = \boldsymbol{\xi}^TB_0\boldsymbol{\xi} \ge 0 \quad (\boldsymbol{\xi}\neq 0 \text{ ならば真不等号}).$$

ここで $\tilde{P}$ が正則であることに注意すると $\boldsymbol{\xi}\neq 0 \Leftrightarrow \boldsymbol{z}\neq 0$ であるから

$$\boldsymbol{z}\neq 0 \Rightarrow \boldsymbol{z}^TI_0\boldsymbol{z} > 0.$$

ゆえに I_0 は正定値であることが分かった。

したがって実対称正定値行列 F_0 で $F_0^2 = I_0$ を満たすものが存在する。

$$\frac{\boldsymbol{z}^T\boldsymbol{z}}{\boldsymbol{z}\tilde{P}B_0\tilde{P}^T\boldsymbol{z}} = \frac{\boldsymbol{z}^T\boldsymbol{z}}{\boldsymbol{z}I_0\boldsymbol{z}} = \frac{(\boldsymbol{z},\boldsymbol{z})}{(I_0\boldsymbol{z},\boldsymbol{z})} = \frac{(\boldsymbol{z},\boldsymbol{z})}{(F_0^2\boldsymbol{z},\boldsymbol{z})} = \frac{(\boldsymbol{z},\boldsymbol{z})}{(I_0\boldsymbol{z},\boldsymbol{z})} = \frac{(\boldsymbol{z},\boldsymbol{z})}{(F_0\boldsymbol{z},F_0\boldsymbol{z})}$$

であるが、$\boldsymbol{\eta} := F_0\boldsymbol{z}$ とおくと、

$$\frac{\boldsymbol{z}^T\boldsymbol{z}}{\boldsymbol{z}\tilde{P}B_0\tilde{P}^T\boldsymbol{z}} = \frac{(F_0^{-1}\boldsymbol{\eta},F_0^{-1}\boldsymbol{\eta})}{(\boldsymbol{\eta},\boldsymbol{\eta})} = \frac{((F_0^{-1})^2\boldsymbol{\eta},\boldsymbol{\eta})}{(\boldsymbol{\eta},\boldsymbol{\eta})} = \frac{(I_0^{-1}\boldsymbol{\eta},\boldsymbol{\eta})}{(\boldsymbol{\eta},\boldsymbol{\eta})}.$$

よって

$$\begin{aligned}
\sup_{0\neq\boldsymbol{z}\in\mathbf{R}^n}\left|\frac{\boldsymbol{z}^T\tilde{P}A_0\tilde{P}^T\boldsymbol{z}}{\boldsymbol{z}^T\boldsymbol{z}}\right| \sup_{0\neq\boldsymbol{z}\in\mathbf{R}^n}\left|\frac{\boldsymbol{z}^T\boldsymbol{z}}{\boldsymbol{z}^T\tilde{P}B_0\tilde{P}^T\boldsymbol{z}}\right| &= \sup_{0\neq\boldsymbol{z}\in\mathbf{R}^n}\left|\frac{\boldsymbol{z}^T\tilde{P}A_0\tilde{P}^T\boldsymbol{z}}{\boldsymbol{z}^T\boldsymbol{z}}\right| \sup_{0\neq\boldsymbol{\eta}\in\mathbf{R}^n}\left|\frac{\boldsymbol{\eta}^T(\tilde{P}B_0\tilde{P}^T)^{-1}\boldsymbol{\eta}}{\boldsymbol{\eta}^T\boldsymbol{\eta}}\right| \\
&= r\left(\tilde{P}A_0\tilde{P}^T\right) r\left((\tilde{P}B_0\tilde{P}^T)^{-1}\right) \\
&\leq \|\tilde{P}A_0\tilde{P}^T\|_\infty \|(\tilde{P}B_0\tilde{P}^T)^{-1}\|_\infty \\
&= \lambda_1\lambda_2.
\end{aligned}$$

よって結論が導かれる。□

6.2.5 対称行列の正定値性判定法

最後に提案する一般化 Rump 法の前に、アルゴリズム中で用いる実対称行列の正定値性を判定する手法を述べる。次のアルゴリズム VPD は区間演算による丸め誤差の考慮を行っているため、正定値性が判定された場合には数学的な厳密性を保証する。またあわせて最小固有値の下界を得ることができる。以下、正定値性を判定する区間行列を X とする。

アルゴリズム：VPD(正定値判定法)

Step 1. X の最小固有値 $\tilde{\rho}$ を求める ($floating$). $\tilde{\rho} \leq 0$ の場合停止。

Step 2. ある微小な数 $0 < \delta \ll 1$ により $\rho := (1-\delta)\tilde{\rho} > 0$ とする ($interval$).

Step 3. $Y = X - \rho I_n$ を計算 ($interval$).

Step 4. Y を Cholesky 分解 ($Y \approx \tilde{C}\tilde{C}^T$) する ($floating$). Cholesky 分解が失敗する場合は δ の値を大きくして Step 2 に戻るか停止。

Step 5. $Z := \tilde{C}\tilde{C}^T - Y, \quad \lambda := \max\limits_{1\leq i\leq n}\sum\limits_{j=1}^{n}|z_{ij}|$ を計算 ($interval$).

Step 6. $\rho - \lambda > 0$ ならば終了。そうでない場合は δ の値を大きくして Step 2 に戻るか停止。

Theorem 6.5
アルゴリズム VPD の Step 6 で $\rho - \lambda > 0$ が成立するならば、X に含まれるすべての実対称行列は正定値である。また $\rho - \lambda$ は X の最小固有値を与える。

証明
X_0 を X に含まれる任意の実対称行列とする。$\boldsymbol{x}^T\boldsymbol{x} = 1$ となる任意の $\boldsymbol{x} \in \mathbf{R}^n$ に対し $\tilde{C}\tilde{C}^T - (X_0 - \rho I_n)$ は実対称行列であるので、

$$|\boldsymbol{x}^T(\tilde{C}\tilde{C}^T - (X_0 - \rho I_n))\boldsymbol{x}| \leq \lambda$$

となる。ここで $\tilde{C}\tilde{C}^T$ は正定値である。なぜなら、$\tilde{C}$ は Cholesky 分解されているので対角成分はすべて 0 ではない。よって $\tilde{C}$ が下三角行列であることを考慮すると $\tilde{C}$ は正則である。よって

$$\boldsymbol{x} \neq 0 \Rightarrow \tilde{C}^T\boldsymbol{x} \neq 0$$

が成り立つ。0 でない任意の $\boldsymbol{x} \in \mathbf{R}^n$ に対して

$$(\tilde{C}\tilde{C}^T\boldsymbol{x}, \boldsymbol{x}) = (\tilde{C}^T\boldsymbol{x}, \tilde{C}^T\boldsymbol{x}) > 0$$

となる。$\tilde{C}\tilde{C}^T$ が正定値より

$$-\boldsymbol{x}^T(X_0 - \rho I_n)\boldsymbol{x} \quad \leq \quad \lambda.$$

従って

$$\boldsymbol{x}^T X_0 \boldsymbol{x} \geq \rho - \lambda$$

となり結論が導かれる。□

6.2.6 一般化 Rump 法

次の一般化 Rump 法 (generalized Rump's method) は Rump の方法をもとに区間演算による Cholesky 分解を回避した方法である。

アルゴリズム：G-Rump(一般化 Rump 法)

Step 1. (6.2) の γ の近似値 $\tilde{\beta}$ を計算 ($floating$).

Step 2. ある微小な数 $0 < \delta \ll 1$ により $\beta := (1 + \delta)\tilde{\beta}$ とする ($interval$).

Step 3. $X_1 := -A + \beta B, \quad X_2 := A + \beta B$ を計算 ($interval$).

Step 4. VPD により X_1, X_2 の正定値性を判定。ともに正定値ならば終了。正定値性が得られなかった場合は δ の値を大きくして Step 2 に戻るか停止。

Theorem 6.6
アルゴリズム G-RUMP の Step 4 で X_1, X_2 の正定値性が判定されたならば、β が (6.2) の 1 つの上界を与える。

証明
A_0 を A に含まれる任意の実対称行列、B_0 を B に含まれる任意の実対称正定値行列とする。X_1, X_2 に含まれる任意の実対称行列は正定値であるので、$\boldsymbol{x}^T\boldsymbol{x} = 1$ となる任意の $\boldsymbol{x} \in \mathbf{R}^n$ に対して

$$\boldsymbol{x}^T(-A_0 + \beta B_0)\boldsymbol{x} > 0, \quad \boldsymbol{x}^T(A_0 + \beta B_0)\boldsymbol{x} > 0.$$

すなわち

$$-\beta \boldsymbol{x}^T B_0 \boldsymbol{x} < \boldsymbol{x}^T A_0 \boldsymbol{x} < \beta \boldsymbol{x}^T B_0 \boldsymbol{x}$$

が成立する。B_0 が正定値より $\boldsymbol{x}^T B_0 \boldsymbol{x}$ なので

$$-\beta < \frac{\boldsymbol{x}^T A_0 \boldsymbol{x}}{\boldsymbol{x}^T B_0 \boldsymbol{x}} < \beta.$$

よって

$$\sup_{\boldsymbol{x}^T \boldsymbol{x}=1} \left| \frac{\boldsymbol{x}^T A_0 \boldsymbol{x}}{\boldsymbol{x}^T B_0 \boldsymbol{x}} \right| \leq \beta$$

を得る。□

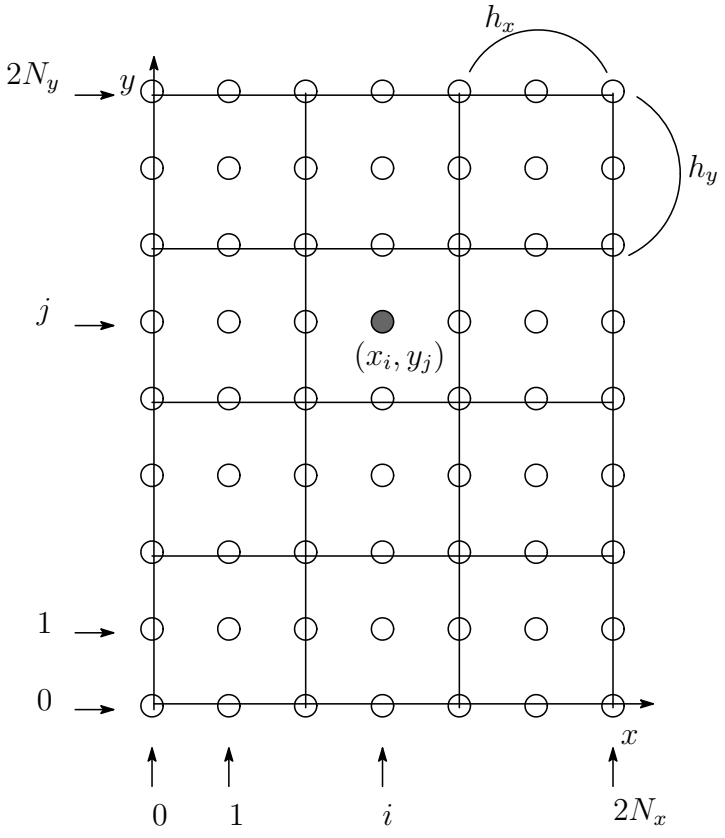
一般化 Rump 法は区間演算による Cholesky 分解および逆行列の区間評価が必要ないことから、計算コストの面で優れている。ただし VPD とあわせて 2 つのパラメータ δ は慎重に選ぶ必要がある。また、A が正定値の場合は、Rump の方法と同様アルゴリズム中の X_2 に関する手順は省略可能である。

7　アプリオリ誤差評価の数値実験

$\Omega = (0, L_x) \times (0, L_y)$ の矩形領域とする。

7.1　基底関数の作成

$X_h^*,\ X_h,\ Y_h$ の次元をそれぞれ $\hat{n},\ n,\ m$ とする。X_h^* の基底関数 $\{\hat{\phi}_j\}_{1 \leq j \leq \hat{n}}$, X_h の基底関数 $\{\phi_j\}_{1 \leq j \leq n}$, Y_h の基底関数 $\{\psi_j\}_{1 \leq j \leq m}$ を以下のようにして定める。



領域Ωは各辺がx軸とy軸に平行な長方形領域とする。Ωをy軸に平行な直線とx軸に平行な直線で等分割する。Ωのx方向の分割数をN_x, y方向の分割数をN_yとし、Ωのx方向の長さをL_x, y方向の長さをL_yとする。さらに、$h_x = L_x/N_x$, $h_y = L_y/N_y$とする。また、図のように(x_i, y_j)を定める。要素は$[x_i, x_{i+2}] \times [y_j, y_{j+2}]$ $(i = 0, 2, \ldots, 2N_x - 2,\ j = 0, 2, \ldots, 2N_y - 2)$の長方形要素である。

まずX_h^*を、その基底関数を以下のように定めることで定義する。$\hat{\phi}_{ij}(x, y)$ $(i = 0, 1, \ldots, 2N_x,$ $j = 0, 1, \ldots, 2N_y)$を次のように定める。

$$
\begin{cases}
(x_i, y_j) \text{を含むそれぞれの要素内で} \\
\qquad \begin{cases} \hat{\phi}_{ij}(x_p, y_q) = \delta_{ip}\delta_{jq}, \\ \hat{\phi}_{ij}(x, y) \text{は双2次。} \end{cases} \\
\\
(x_i, y_j) \text{を含まない要素内で } \hat{\phi}_{ij}(x, y) \equiv 0.
\end{cases}
$$

次に、$(i, j) \ \rightarrow \ i(2N_y + 1) + (j + 1)$と1次元的な番号付けを行う。すると、$\hat{\phi}_1$, $\hat{\phi}_2$,..., $\hat{\phi}_{\hat{n}}$が定まる。$\hat{n} = (2N_x + 1)(2N_y + 1)$である。

X_hは、境界上の節点に対応する基底関数を除いた$\hat{\phi}_{ij}$ $(1 \le i \le N_x - 1,\ 1 \le j \le N_y - 1)$で張られた空間とする。$X_h^*$の基底関数から境界上の節点に対応する基底関数を削除すればよい。$\hat{\phi}_{ij}(x, y)$ $(i = 1, 2, \ldots, 2N_x - 1,\ j = 1, 2, \ldots, 2N_y - 1)$で$(i, j) \ \rightarrow \ (i-1)(2N_y+1)+j$と1次元的な番号付けを行う。すると、$\phi_1$, ϕ_2,..., $\phi_{(2N_x-1)(2N_y-1)}$が定まる。$n = (2N_x - 1)(2N_y - 1)$である。

区分双1次関数でΩでの平均が0であるもの全体をY_hとする。$\tilde{\psi}_{ij}(x, y)$ $(i = 0, 2, \ldots, 2N_x,$ $j = 0, 2, \ldots, 2N_y)$を次のように定める。

$$
\begin{cases}
(x_i, y_j) \text{を含むそれぞれの要素内で} \\
\qquad \begin{cases} \tilde{\psi}_{ij}(x_p, y_q) = \delta_{ip}\delta_{jq}, \\ \tilde{\psi}_{ij}(x, y) \text{は双1次。} \end{cases} \\
\\
(x_i, y_j) \text{を含まれない要素内で } \tilde{\psi}_{ij}(x, y) \equiv 0.
\end{cases}
$$

次に、$(i, j) \ \rightarrow \ \dfrac{i(N_y + 1)}{2} + \dfrac{j}{2} + 1$と1次元的な番号付けを行う。すると、$\tilde{\psi}_1$, $\tilde{\psi}_2$,..., $\tilde{\psi}_{(N_x+1)(N_y+1)}$が定まる。$\tilde{m} = (N_x + 1)(N_y + 1)$とする。これを$Y_h$の基底関数になるように調整する。

$$
\begin{aligned}
\beta_l &:= \frac{\displaystyle\int_\Omega \tilde{\psi}_l}{\displaystyle\int_\Omega \tilde{\psi}_{\tilde{m}}} \qquad (l = 1, 2, \ldots, \tilde{m} - 1), \\
\psi_l &:= \tilde{\psi}_l - \beta_l \tilde{\psi}_{\tilde{m}} \quad (l = 1, 2, \ldots, \tilde{m} - 1)
\end{aligned}
$$

とする。すると、

$$
\int_\Omega \psi_l = \int_\Omega \tilde{\psi}_l - \frac{\displaystyle\int_\Omega \tilde{\psi}_l}{\displaystyle\int_\Omega \tilde{\psi}_{\tilde{m}}} \int_\Omega \tilde{\psi}_{\tilde{m}} = 0
$$

より $\psi_l \in Y_h$ である。また、$i = 0, 1, \ldots, 2N_x,\ j = 0, 1, \ldots, 2N_y,\ (i, j) \neq (2N_x, 2N_y)$ であるような (i, j) に対して、

$$\psi_{ij}(x_p, y_q) = \tilde{\psi}_{ij}(x_p, y_q) - \beta_{ij}\tilde{\psi}_{2N_x,2Ny}(x_p, y_q) = \tilde{\psi}_{ij}(x_p, y_q) = \delta_{ip}\delta_{jq}, \quad ((p, q) \neq (2N_x, 2N_y)).$$

よって ψ_l は 1 次独立である。ゆえに $\{\psi_l\}_{1 \leq l \leq \tilde{m}-1}$ は Y_h の基底となる。$m = (N_x+1)(N_y+1)-1$ である。また、

$$\beta_l = \begin{cases} 1 & (\text{節点 } l \text{ が境界の角にあるとき}), \\ 2 & (\text{節点 } l \text{ が境界の辺上にあるとき}), \\ 4 & (\text{それ以外}) \end{cases}$$

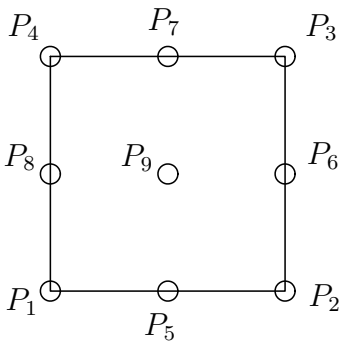
となる。

7.2 行列の作成

基底関数を用いて以下のように行列を定義する。

$$\begin{cases}
\hat{L} = ((\hat{\phi}_i, \hat{\phi}_j)) \in \mathbf{R}^{\hat{n}\times\hat{n}}, & L = ((\phi_i, \phi_j)) \in \mathbf{R}^{n\times n}, \\
K^x = \left(\left(\dfrac{\partial\phi_i}{\partial x}, \hat{\phi}_j\right)\right) \in \mathbf{R}^{n\times\hat{n}}, & K^y = \left(\left(\dfrac{\partial\phi_i}{\partial y}, \hat{\phi}_j\right)\right) \in \mathbf{R}^{n\times\hat{n}}, \\
\hat{D}^{xx} = \left(\left(\dfrac{\partial\hat{\phi}_i}{\partial x}, \dfrac{\partial\hat{\phi}_j}{\partial x}\right)\right) \in \mathbf{R}^{\hat{n}\times\hat{n}}, & \hat{D}^{xy} = \left(\left(\dfrac{\partial\hat{\phi}_i}{\partial x}, \dfrac{\partial\hat{\phi}_j}{\partial y}\right)\right) \in \mathbf{R}^{\hat{n}\times\hat{n}}, \quad \hat{D}^{yy} = \left(\left(\dfrac{\partial\hat{\phi}_i}{\partial y}, \dfrac{\partial\hat{\phi}_j}{\partial y}\right)\right) \in \mathbf{R}^{\hat{n}\times\hat{n}}, \\
\tilde{D} = ((\nabla\psi_i, \nabla\psi_j)) \in \mathbf{R}^{m\times m}, \\
\hat{F}^{xx} = \left(\left(\dfrac{\partial\hat{\phi}_i}{\partial x}, \dfrac{\partial\psi_j}{\partial x}\right)\right) \in \mathbf{R}^{\hat{n}\times m}, & \hat{F}^{xy} = \left(\left(\dfrac{\partial\hat{\phi}_i}{\partial x}, \dfrac{\partial\psi_j}{\partial y}\right)\right) \in \mathbf{R}^{\hat{n}\times m}, \\
\hat{F}^{yx} = \left(\left(\dfrac{\partial\hat{\phi}_i}{\partial y}, \dfrac{\partial\psi_j}{\partial x}\right)\right) \in \mathbf{R}^{\hat{n}\times m}, & \hat{F}^{yy} = \left(\left(\dfrac{\partial\hat{\phi}_i}{\partial y}, \dfrac{\partial\psi_j}{\partial y}\right)\right) \in \mathbf{R}^{\hat{n}\times m}, \\
D_0 = ((\nabla\phi_i, \nabla\phi_j)) \in \mathbf{R}^{n\times n}, \\
D^{xx} = \left(\left(\dfrac{\partial\phi_i}{\partial x}, \dfrac{\partial\phi_j}{\partial x}\right)\right) \in \mathbf{R}^{n\times n}, & D^{xy} = \left(\left(\dfrac{\partial\phi_i}{\partial x}, \dfrac{\partial\phi_j}{\partial y}\right)\right) \in \mathbf{R}^{n\times n}, \quad D^{yy} = \left(\left(\dfrac{\partial\phi_i}{\partial y}, \dfrac{\partial\phi_j}{\partial y}\right)\right) \in \mathbf{R}^{n\times n}, \\
E_x = \left(\left(\psi_i, \dfrac{\partial\phi_j}{\partial x}\right)\right) \in \mathbf{R}^{m\times n}, & E_y = \left(\left(\psi_i, \dfrac{\partial\phi_j}{\partial y}\right)\right) \in \mathbf{R}^{m\times n}.
\end{cases}$$

これらの行列を作成するプログラムを書くために、Mathematica を用いて要素係数行列を求めた (プログラムは 9.2.1 節参照)。



要素係数行列の局所節点番号は上図の通りである。上の各行列の要素係数行列を添え字 $^{(0)}$ を付加して表すことにする。求めた要素係数行列から上の各行列を組み立てていく。ただし ψ を用いる行列では、まず要素係数行列で ψ_{ij} の代わりに $\tilde{\psi}_{ij}$ を用いて行列を組み立てる。そして最後に以下の操作を行うことにする。

$$\hat{F}^{xx}_{ij} = \left(\frac{\partial\hat{\phi}_i}{\partial x}, \frac{\partial\psi_j}{\partial x}\right) = \left(\frac{\partial\hat{\phi}_i}{\partial x}, \frac{\partial\tilde{\psi}_j}{\partial x} - \beta_j\frac{\partial\tilde{\psi}_m}{\partial x}\right) = \left(\frac{\partial\hat{\phi}_i}{\partial x}, \frac{\partial\tilde{\psi}_j}{\partial x}\right) - \beta_j\left(\frac{\partial\hat{\phi}_i}{\partial x}, \frac{\partial\tilde{\psi}_m}{\partial x}\right).$$

同様に、

$$\begin{aligned}
\hat{F}^{xy}_{ij} &= \left(\frac{\partial\hat{\phi}_i}{\partial x}, \frac{\partial\tilde{\psi}_j}{\partial y}\right) - \beta_j\left(\frac{\partial\hat{\phi}_i}{\partial x}, \frac{\partial\tilde{\psi}_m}{\partial y}\right), \\
\hat{F}^{yx}_{ij} &= \left(\frac{\partial\hat{\phi}_i}{\partial y}, \frac{\partial\tilde{\psi}_j}{\partial x}\right) - \beta_j\left(\frac{\partial\hat{\phi}_i}{\partial y}, \frac{\partial\tilde{\psi}_m}{\partial x}\right), \\
\hat{F}^{yy}_{ij} &= \left(\frac{\partial\hat{\phi}_i}{\partial y}, \frac{\partial\tilde{\psi}_j}{\partial y}\right) - \beta_j\left(\frac{\partial\hat{\phi}_i}{\partial y}, \frac{\partial\tilde{\psi}_m}{\partial y}\right).
\end{aligned}$$

$$\begin{aligned}
\tilde{D}_{ij} &= (\nabla\psi_i, \nabla\psi_j) \\
&= (\nabla\tilde{\psi}_i - \beta_i\nabla\tilde{\psi}_m, \nabla\tilde{\psi}_j - \beta_j\nabla\tilde{\psi}_m) \\
&= (\nabla\tilde{\psi}_i, \nabla\tilde{\psi}_j) - \beta_i(\nabla\tilde{\psi}_m, \nabla\tilde{\psi}_j) - \beta_j(\nabla\tilde{\psi}_m, \nabla\tilde{\psi}_i) + \beta_i\beta_j(\nabla\tilde{\psi}_m, \nabla\tilde{\psi}_m).
\end{aligned}$$

$$E^x_{ij} = \left(\psi_i, \frac{\partial\phi_j}{\partial x}\right) = \left(\tilde{\psi}_i - \beta_i\tilde{\psi}_m, \frac{\partial\phi_j}{\partial x}\right) = \left(\tilde{\psi}_i - \frac{\partial\phi_j}{\partial x}\right) - \beta_i\left(\tilde{\psi}_m - \frac{\partial\phi_j}{\partial x}\right).$$

同様に、

$$E^y_{ij} = \left(\tilde{\psi}_i - \frac{\partial\phi_j}{\partial y}\right) - \beta_i\left(\tilde{\psi}_m - \frac{\partial\phi_j}{\partial y}\right).$$

以下に求めた要素係数行列を示す。

$$\hat{L}^{(0)} = \frac{h_x h_y}{900}\begin{pmatrix}
16 & -4 & 1 & -4 & 8 & -2 & -2 & 8 & 4 \\
-4 & 16 & -4 & 1 & 8 & 8 & -2 & -2 & 4 \\
1 & -4 & 16 & -4 & -2 & 8 & 8 & -2 & 4 \\
-4 & 1 & -4 & 16 & -2 & -2 & 8 & 8 & 4 \\
8 & 8 & -2 & -2 & 64 & 4 & -16 & 4 & 32 \\
-2 & 8 & 8 & -2 & 4 & 64 & 4 & -16 & 32 \\
-2 & -2 & 8 & 8 & -16 & 4 & 64 & 4 & 32 \\
8 & -2 & -2 & 8 & 4 & -16 & 4 & 64 & 32 \\
4 & 4 & 4 & 4 & 32 & 32 & 32 & 32 & 256
\end{pmatrix},$$

$$K_x^{(0)} = \frac{h_y}{180}\begin{pmatrix} -12 & 4 & -1 & 3 & -16 & 2 & 4 & -6 & -8 \\ -4 & 12 & -3 & 1 & 16 & 6 & -4 & -2 & 8 \\ 1 & -3 & 12 & -4 & -4 & 6 & 16 & -2 & 8 \\ 3 & -1 & 4 & -12 & 4 & 2 & -16 & -6 & -8 \\ 16 & -16 & 4 & -4 & 0 & -8 & 0 & 8 & 0 \\ -2 & 6 & 6 & -2 & 8 & 48 & 8 & -16 & 64 \\ -4 & 4 & -16 & 16 & 0 & -8 & 0 & 8 & 0 \\ -6 & 2 & 2 & -6 & -8 & 16 & -8 & -48 & -64 \\ 8 & -8 & -8 & 8 & 0 & -64 & 0 & 64 & 0 \end{pmatrix},$$

$$K_y^{(0)} = \frac{h_x}{180}\begin{pmatrix} -12 & 3 & -1 & 4 & -6 & 4 & 2 & -16 & -8 \\ 3 & -12 & 4 & -1 & -6 & -16 & 2 & 4 & -8 \\ 1 & -4 & 12 & -3 & -2 & 16 & 6 & -4 & 8 \\ -4 & 1 & -3 & 12 & -2 & -4 & 6 & 16 & 8 \\ -6 & -6 & 2 & 2 & -48 & -8 & 16 & -8 & -64 \\ -4 & 16 & -16 & 4 & 8 & 0 & -8 & 0 & 0 \\ -2 & -2 & 6 & 6 & -16 & 8 & 48 & 8 & 64 \\ 16 & -4 & 4 & -16 & 8 & 0 & -8 & 0 & 0 \\ 8 & 8 & -8 & -8 & 64 & 0 & -64 & 0 & 0 \end{pmatrix},$$

$$\hat{D}_{xx}^{(0)} = \frac{h_y}{90h_x}\begin{pmatrix} 28 & 4 & -1 & -7 & -32 & 2 & 8 & 14 & -16 \\ 4 & 28 & -7 & -1 & -32 & 14 & 8 & 2 & -16 \\ -1 & -7 & 28 & 4 & 8 & 14 & -32 & 2 & -16 \\ -7 & -1 & 4 & 28 & 8 & 2 & -32 & 14 & -16 \\ -32 & -32 & 8 & 8 & 64 & -16 & -16 & -16 & 32 \\ 2 & 14 & 14 & 2 & -16 & 112 & -16 & 16 & -128 \\ 8 & 8 & -32 & -32 & -16 & -16 & 64 & -16 & 32 \\ 14 & 2 & 2 & 14 & -16 & 16 & -16 & 112 & -128 \\ -16 & -16 & -16 & -16 & 32 & -128 & 32 & -128 & 256 \end{pmatrix},$$

$$\hat{D}_{xy}^{(0)} = \frac{1}{36}\begin{pmatrix} 9 & -3 & -1 & 3 & 12 & 4 & 4 & -12 & -16 \\ 3 & -9 & -3 & 1 & -12 & 12 & -4 & -4 & 16 \\ -1 & 3 & 9 & -3 & 4 & -12 & 12 & 4 & -16 \\ -3 & 1 & 3 & -9 & -4 & -4 & -12 & 12 & 16 \\ -12 & 12 & 4 & -4 & 0 & -16 & 0 & 16 & 0 \\ 4 & -12 & 12 & -4 & -16 & 0 & 16 & 0 & 0 \\ 4 & -4 & -12 & 12 & 0 & 16 & 0 & -16 & 0 \\ 12 & -4 & 4 & -12 & 16 & 0 & -16 & 0 & 0 \\ -16 & 16 & -16 & 16 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\hat{D}^{(0)}_{yy} = \frac{h_x}{90h_y}\begin{pmatrix} 28 & -7 & -1 & 4 & 14 & 8 & 2 & -32 & -16 \\ -7 & 28 & 4 & -1 & 14 & -32 & 2 & 8 & -16 \\ -1 & 4 & 28 & -7 & 2 & -32 & 14 & 8 & -16 \\ 4 & -1 & -7 & 28 & 2 & 8 & 14 & -32 & -16 \\ 14 & 14 & 2 & 2 & 112 & -16 & 16 & -16 & -128 \\ 8 & -32 & -32 & 8 & -16 & 64 & -16 & -16 & 32 \\ 2 & 2 & 14 & 14 & 16 & -16 & 112 & -16 & -128 \\ -32 & 8 & 8 & -32 & -16 & -16 & -16 & 64 & 32 \\ -16 & -16 & -16 & -16 & -128 & 32 & -128 & 32 & 256 \end{pmatrix},$$

$$\tilde{D}^{(0)} = \frac{1}{6h_xh_y}\begin{pmatrix} 2h_x^2+2h_y^2 & h_x^2-2h_y^2 & -h_x^2-h_y^2 & -2h_x^2+h_y^2 \\ h_x^2-2h_y^2 & 2h_x^2+2h_y^2 & -2h_x^2+h_y^2 & -h_x^2-h_y^2 \\ -h_x^2-h_y^2 & -2h_x^2+h_y^2 & 2h_x^2+2h_y^2 & h_x^2-2h_y^2 \\ -2h_x^2+h_y^2 & -h_x^2-h_y^2 & h_x^2-2h_y^2 & 2h_x^2+2h_y^2 \end{pmatrix},$$

$$\hat{F}^{(0)}_{xx} = \frac{h_y}{6h_x}\begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 \\ -2 & 2 & 2 & -2 \\ 0 & 0 & 0 & 0 \\ 2 & -2 & -2 & 2 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \hat{F}^{(0)}_{xy} = \frac{1}{36}\begin{pmatrix} 5 & 1 & -1 & -5 \\ -1 & -5 & 5 & 1 \\ -1 & -5 & 5 & 1 \\ 5 & 1 & -1 & -5 \\ -4 & 4 & -4 & 4 \\ -4 & -20 & 20 & 4 \\ -4 & 4 & -4 & 4 \\ 20 & 4 & -4 & -20 \\ -16 & 16 & -16 & 16 \end{pmatrix},$$

$$\hat{F}^{(0)}_{yx} = \frac{1}{36}\begin{pmatrix} 5 & -5 & -1 & 1 \\ 5 & -5 & -1 & 1 \\ -1 & 1 & 5 & -5 \\ -1 & 1 & 5 & -5 \\ 20 & -20 & -4 & 4 \\ -4 & 4 & -4 & 4 \\ -4 & 4 & 20 & -20 \\ -4 & 4 & -4 & 4 \\ -16 & 16 & -16 & 16 \end{pmatrix}, \quad \hat{F}^{(0)}_{yy} = \frac{h_x}{6h_y}\begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 2 & 2 & -2 & -2 \\ 0 & 0 & 0 & 0 \\ -2 & -2 & 2 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$
D_0^{(0)} = \frac{h_x}{90h_y}\begin{pmatrix}
28 & -7 & -1 & 4 & 14 & 8 & 2 & -32 & -16\\
-7 & 28 & 4 & -1 & 14 & -32 & 2 & 8 & -16\\
-1 & 4 & 28 & -7 & 2 & -32 & 14 & 8 & -16\\
4 & -1 & -7 & 28 & 2 & 8 & 14 & -32 & -16\\
14 & 14 & 2 & 2 & 112 & -16 & 16 & -16 & -128\\
8 & -32 & -32 & 8 & -16 & 64 & -16 & -16 & 32\\
2 & 2 & 14 & 14 & 16 & -16 & 112 & -16 & -128\\
-32 & 8 & 8 & -32 & -16 & -16 & -16 & 64 & 32\\
-16 & -16 & -16 & -16 & -128 & 32 & -128 & 32 & 256
\end{pmatrix}
$$

$$
+ \frac{h_y}{90h_x}\begin{pmatrix}
28 & 4 & -1 & -7 & -32 & 2 & 8 & 14 & -16\\
4 & 28 & -7 & -1 & -32 & 14 & 8 & 2 & -16\\
-1 & -7 & 28 & 4 & 8 & 14 & -32 & 2 & -16\\
-7 & -1 & 4 & 28 & 8 & 2 & -32 & 14 & -16\\
-32 & -32 & 8 & 8 & 64 & -16 & -16 & -16 & 32\\
2 & 14 & 14 & 2 & -16 & 112 & -16 & 16 & -128\\
8 & 8 & -32 & -32 & -16 & -16 & 64 & -16 & 32\\
14 & 2 & 2 & 14 & -16 & 16 & -16 & 112 & -128\\
-16 & -16 & -16 & -16 & 32 & -128 & 32 & -128 & 256
\end{pmatrix},
$$

$$
E_x^{(0)} = \frac{h_y}{36}\begin{pmatrix}
-5 & 1 & 0 & 0 & 4 & 2 & 0 & -10 & 8\\
-1 & 5 & 0 & 0 & -4 & 10 & 0 & -2 & -8\\
0 & 0 & 5 & -1 & 0 & 10 & -4 & -2 & -8\\
0 & 0 & 1 & -5 & 0 & 2 & 4 & -10 & 8
\end{pmatrix},
$$

$$
E_y^{(0)} = \frac{h_x}{36}\begin{pmatrix}
-5 & 0 & 0 & 1 & -10 & 0 & 2 & 4 & 8\\
0 & -5 & 1 & 0 & -10 & 4 & 2 & 0 & 8\\
0 & -1 & 5 & 0 & -2 & -4 & 10 & 0 & -8\\
-1 & 0 & 0 & 5 & -2 & 0 & 10 & -4 & -8
\end{pmatrix}.
$$

7.3 実験結果

9.2 節のプログラムを用いて、$|u-u_h|_1$ と $|p-p_h|_0$ をアプリオリ誤差評価する際に表れる定数を、すなわち K_1, K_2, K_3, K_4 の上界と

$$
\begin{aligned}
C_1(h) &= \sqrt{(\nu K_1 + C_0 h K_2 + K_3)^2 + (C_0 h)^2},\\
C_2(h) &= \sqrt{(C_0 h K_4 + K_3)^2 + (C_0 h)^2}
\end{aligned}
$$

の上界を評価する。プログラムは MATLAB で行い、区間演算には S.M.Rump の INTLAB を用いた。

実験は $\Omega = (0,1)\times(0,1)$, $\nu = 1$, $N = N_x = N_y$ で行った。$C_0 = 1/2\pi$ である。一般化固有値問題の固有値の絶対値最大の上界を評価するアルゴリズムには、改良近似対角化法と一般化 Rump 法を用いた。一般化 Rump 法のパラメータは $N = 5$ のとき $\delta = 10^{-9}$, $N = 10$ のとき $\delta = 10^{-7}$, $N = 15$ のとき $\delta = 10^{-5}$, VPD のパラメータは N によらず $\delta = 10^{-2}$ である。実験結果を以下に示す。

N	アルゴリズム	K_1	K_2
5	floating	2.532827962940464e-002	1.018179641618650e+000
	ADM-a	2.532827965302695e-002	1.018179643786901e+000
	G-RUMP	2.532827964206877e-002	1.018179642127737e+000
10	floating	1.283934945864276e-002	1.012526112790580e+000
	ADM-a	1.283935140304658e-002	1.012526214174302e+000
	G-RUMP	1.283935010061022e-002	1.012526163416885e+000
15	floating	8.585094955574347e-003	1.012397134212051e+000
	ADM-a	8.585128130973332e-003	1.012398072163981e+000
	G-RUMP	8.585137880941854e-003	1.012402196185060e+000

N	アルゴリズム	K_3	K_4
5	floating	5.100876307940919e-002	1.268294924764598e+000
	ADM-a	5.100876308621370e-002	1.268294925934201e+000
	G-RUMP	5.100876310491358e-002	1.268294925398747e+000
10	floating	2.794083993689706e-002	1.238111241605194e+000
	ADM-a	2.794084060492480e-002	1.238111353254126e+000
	G-RUMP	2.794084133393907e-002	1.238111303510755e+000
15	floating	1.896595841433702e-002	1.238013496098485e+000
	ADM-a	1.896597079889749e-002	1.238014931065163e+000
	G-RUMP	1.896605324389201e-002	1.238019686150490e+000

N	アルゴリズム	$C_1(h)$	$C_2(h)$
5	floating	1.133095679252860e-001	9.676511663795480e-002
	ADM-a	1.133095680207260e-001	9.676511667953829e-002
	G-RUMP	1.133095679774698e-001	9.676511668110199e-002
10	floating	5.907917463015094e-002	5.023388842973298e-002
	ADM-a	5.907917869992103e-002	5.023389074875102e-002
	G-RUMP	5.907917736973382e-002	5.023389068930383e-002
15	floating	3.973571265434737e-002	3.380972592273504e-002
	ADM-a	3.973576615066648e-002	3.380975213793257e-002
	G-RUMP	3.973589716649834e-002	3.380987832209992e-002

$C_1(h)$ と $C_2(h)$ の厳密な上界を得ることができた。interval では floating の場合よりも値が大きくなっていることが確認できる。また、$N = 5, 10, 15$ の場合は $C_1(h) > C_2(h)$ となることがわかる。渡辺、山本、中尾 [10] には $K_3, K_4, C_2(h)$ を計算した結果が載っている。その結果を以下に示す。

N	アルゴリズム	K_3	K_4	$C_2(h)$
5	floating	5.100876307940880-e02	1.268294924764600	9.676511663795448-e02
	ADM-a	5.100876404340230e-02	1.268294979559628	9.676511919541045-e02
	G-RUMP	5.100876310491353e-02	1.268294925398748	9.676511668110258-e02
10	floating	2.794083993689644-e02	1.238111241605190	5.023388842973234-e02
	ADM-a	2.794085315113312-e02	1.238113249056619	5.023393126685157-e02
	G-RUMP	2.794084133393883-e02	1.238111303510752	5.023389068930387-e02
15	floating	1.896595841433506-e02	1.238013496098524	3.380972592273357-e02
	ADM-a	1.898434068678022-e02	1.242378307644777	3.387115806908502-e02
	G-RUMP	1.896605324389127-e02	1.238019686150493	3.380987832209943-e02

この結果と比べると floating と一般化 Rump 法では N によらず 14 〜 15 桁、改良近似対角化法では $N = 5$ のとき 7 〜 8 桁、$N = 10$ のとき 5 〜 6 桁、$N = 15$ のとき 1 〜 3 桁だけ

値があっていることがわかる。また、改良近似対角化法では今回の実験値のほうが良い結果が得られていることもわかる。これは近似対角化行列の選び方による影響と考えられる。

また、$|u-u_h|_1$ と $|p-p_h|_0$ に対するそれぞれのアプリオリ誤差定数

$$C_1^{(1)} = \left(\frac{1}{\nu^2}+\frac{1}{\beta^2}\right)^{\frac{1}{2}} C_1(h), \qquad C_1^{(2)} = \left(\frac{1}{\beta}+\frac{\nu}{\beta^2}\right) C_1(h),$$

$$C_2^{(1)} = \left(\frac{1}{\nu^2}+\frac{1}{\beta^2}\right)^{\frac{1}{2}} C_2(h), \qquad C_2^{(2)} = \left(\frac{1}{\beta}+\frac{\nu}{\beta^2}\right) C_2(h)$$

は以下のようになった。ここで、$\dfrac{1}{\beta} = \sqrt{4+2\sqrt{2}}$ である。

N	アルゴリズム	$C_1^{(1)}$	$C_1^{(2)}$
5	floating	3.170325398744435e-001	1.069818297148940e+000
	ADM-a	3.170325401414779e-001	1.069818298050042e+000
	G-RUMP	3.170325400204502e-001	1.069818297641636e+000
10	floating	1.652995517468810e-001	5.577991616865783e-001
	ADM-a	1.652995631338236e-001	5.577992001115282e-001
	G-RUMP	1.652995594120491e-001	5.577991875524943e-001
15	floating	1.111778478833788e-001	3.751668391843461e-001
	ADM-a	1.111779975624767e-001	3.751673442726807e-001
	G-RUMP	1.111783641359507e-001	3.751685812656087e-001

N	アルゴリズム	$C_2^{(1)}$	$C_2^{(2)}$
5	floating	2.707422794093243e-001	9.136129825620294e-001
	ADM-a	2.707422795256720e-001	9.136129829546416e-001
	G-RUMP	2.707422795300473e-001	9.136129829694059e-001
10	floating	1.405510366710497e-001	4.742859227431048e-001
	ADM-a	1.405510431595060e-001	4.742859446382364e-001
	G-RUMP	1.405510429931768e-001	4.742859440769625e-001
15	floating	9.459733611208575e-002	3.192163210575709e-001
	ADM-a	9.459740946044298e-002	3.192165685697495e-001
	G-RUMP	9.459776251524507e-002	3.192177599442606e-001

floating で $N=2$ から $N=20$ まで求めた結果を対数グラフで図 1 に示す。傾きを調べると、どれも分割数のおよそ 0.98 乗に比例して値が減少していくことが分かる。

また $|u-u_h|_0$ をアプリオリ誤差評価する際に現れる定数

$$\nu C_2^{(1)^2} + 2C_2^{(2)}K_3$$

は以下のようになった。

N	アルゴリズム	$\nu C_2^{(1)^2} + 2C_2^{(2)}K_3$
5	floating	1.665059182073146e-001
	ADM-a	1.665059183228017e-001
	G-RUMP	1.665059183608460e-001
10	floating	4.625848801268404e-002
	ADM-a	4.625849169381160e-002
	G-RUMP	4.625849230721359e-002
15	floating	2.105714294021403e-002
	ADM-a	2.105717411266297e-002
	G-RUMP	2.105733873588829e-002

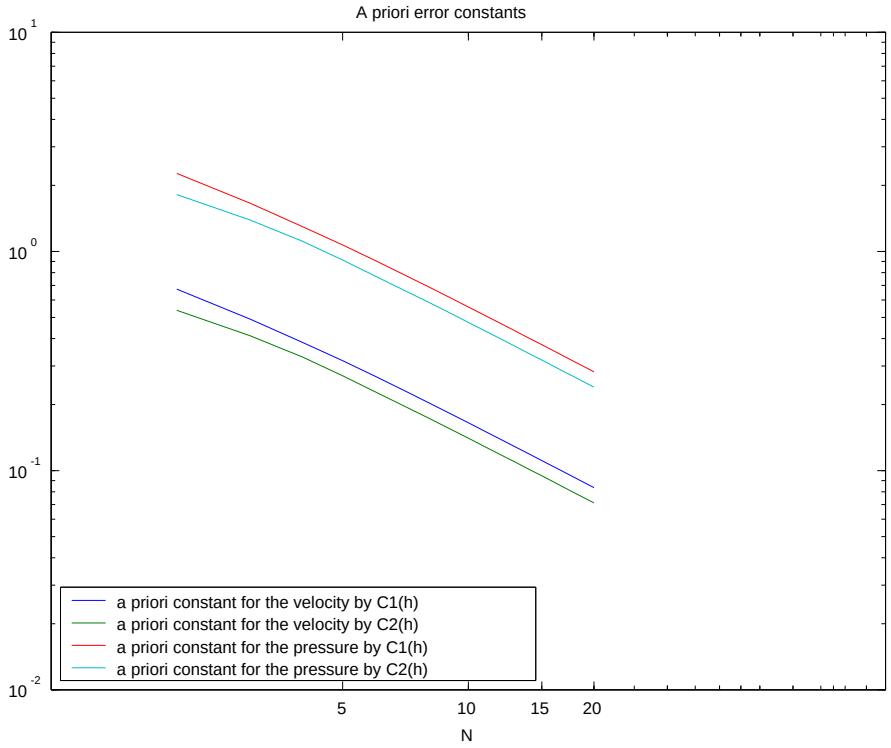


図 1: アプリオリ誤差定数

floating で $N = 2$ から $N = 20$ まで求めた結果を対数グラフで図 2 に示す。傾きを調べると、分割数のおよそ 1.95 乗に比例して値が減少していくことが分かる。floating で $N = 2$ から $N = 20$ まで求めた結果を対数グラフで図 5 に示す。傾きを調べると分割数のおよそ 3.00 乗に比例して値が減少していくことが分かる。

行列 A_1, A_2, A_3, A_4, F を floating で構成するための計算時間を $N = 2$ から $N = 20$ まで求めた結果を対数グラフで図 3 に示す。計算機は Sun Fire V250 を用いた。傾きを調べると分割数の 6 乗程度に比例して計算時間が増えていることがわかる。これは未知数の個数が N^2 に比例して増え、さらに密行列の逆行列を計算する時間が (未知数の個数)^3 に比例することが原因になっていると考えられる。

8　事後誤差評価の数値実験

この節では Theorem 3.1 を用いて事後誤差評価の実験を行う。基底関数や基底関数からの行列の作成方法は前節の通りである。

8.1　事後誤差評価式に現れる定数の評価方法

まず事後誤差評価式に現れる定数

$$C(u_h, p_h) = \nu|\overline{\nabla} u_h - \nabla u_h|_0 + C_0 h|\nu\overline{\triangle} u_h - \nabla p_h + f|_0 + |\operatorname{div} u_h|_0 \tag{8.1}$$

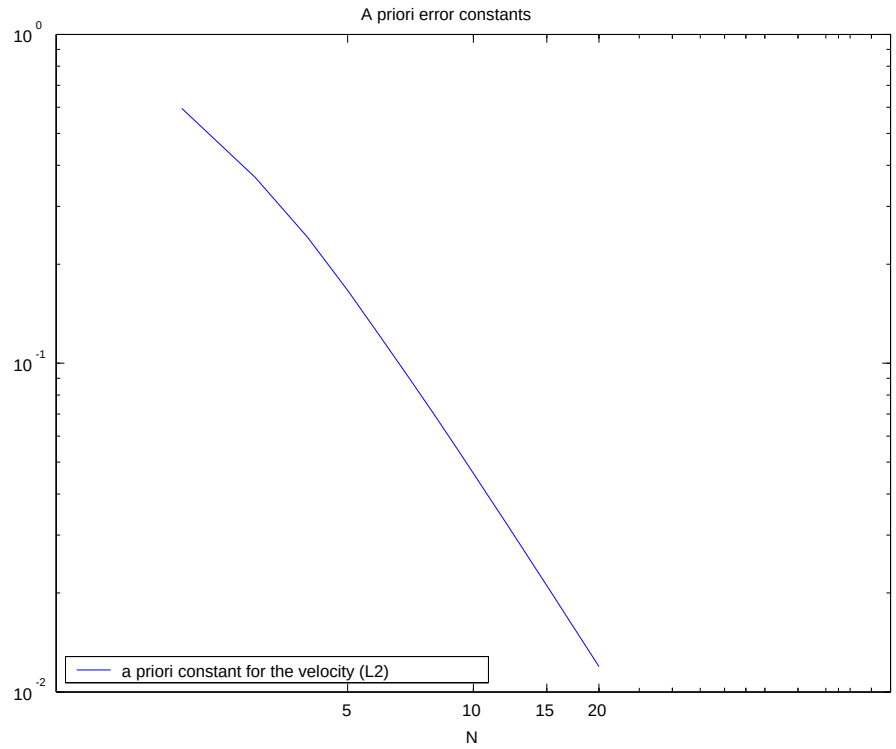


図 2: アプリオリ誤差定数 (L^2)

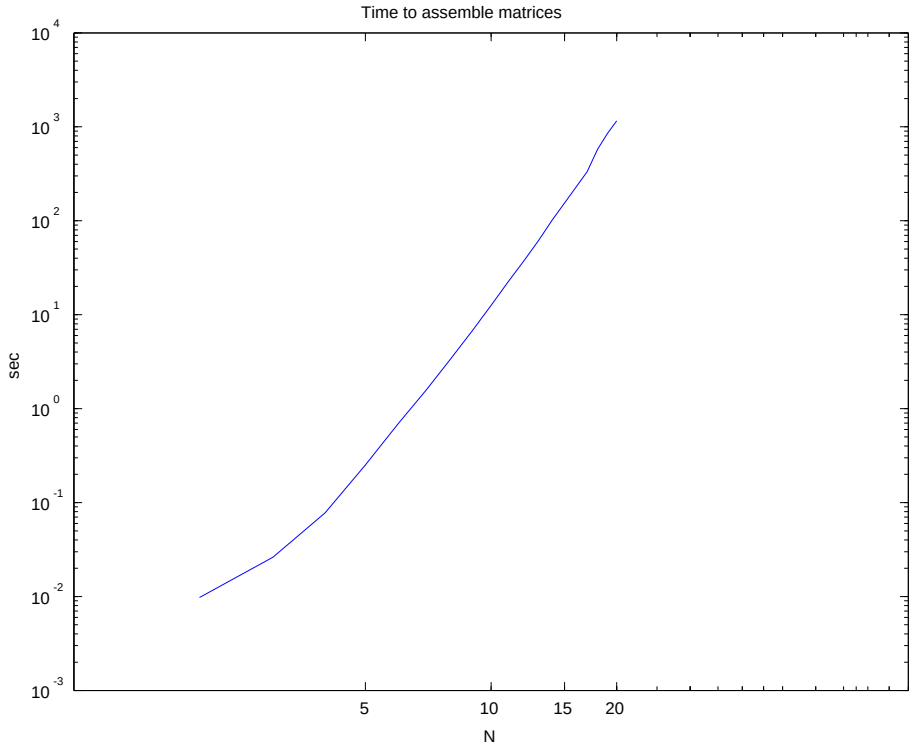


図 3: 行列 A_1, A_2, A_3, A_4, F の floating での計算時間

を求める。(8.1) の第 1 項と第 3 項に現れる $|\overline{\nabla}u_h - \nabla u_h|_0$, $|\operatorname{div} u_h|_0$ は Lemma 4.4, Lemma 4.6 から以下のように計算できる。

$$|\overline{\nabla}u_h - \nabla u_h|_0^2 = \boldsymbol{f}^T A_1 \boldsymbol{f}, \tag{8.2}$$

$$|\operatorname{div} u_h|_0^2 = \boldsymbol{f}^T A_3 \boldsymbol{f}. \tag{8.3}$$

$f = [f_1, f_2]$ が区分双 2 次のとき、$\boldsymbol{f_1} = ((f_1, \phi_1), (f_1, \phi_2), \ldots, (f_1, \phi_n))^T$ は次のようにして求められる。

$$\tilde{\boldsymbol{f}}_{\boldsymbol{1}} = (f_1(P_j))_{j=1}^{\hat{n}}$$

とする。ここで $(P_j)_{j=1}^{\hat{n}}$ は節点上の座標 (x_i, y_j) $(i = 0, 1, \ldots, 2N_x,\ j = 0, 1, \ldots, 2N_y)$ を 1 次元的に番号付けしたものである。

$$(f_1, \hat{\phi}_i) = \sum_{j=1}^{\hat{n}} f_1(P_j)(\hat{\phi}_j, \hat{\phi}_i)$$

より、

$$\hat{\boldsymbol{f}}_1 = \hat{L}\tilde{\boldsymbol{f}}_1$$

を得る。ここで、$\hat{\boldsymbol{f}}_1 = ((f_1, \hat{\phi}_1), (f_1, \hat{\phi}_2), \ldots, (f_1, \hat{\phi}_{\hat{n}}))^T$ である。$\hat{\boldsymbol{f}}_1$ から境界上の節点に対応する番号を除くと $\boldsymbol{f}_1$ を得る。同様にして $\boldsymbol{f}_2$ を得る。このようにして $\boldsymbol{f} = (\boldsymbol{f}_1^T \boldsymbol{f}_2^T)^T$ が求まる。

第 2 項に現れる $|\nu\overline{\triangle}u_h - \nabla p_h + f|_0$ は次のようにして計算する。

$$\begin{aligned}
|\nu\overline{\triangle}u_h - \nabla p_h + f|_0^2 &= (\nu\overline{\triangle}u_h - \nabla p_h + P_0 f, \nu\overline{\triangle}u_h - \nabla p_h + f) \\
&= \nu^2(\overline{\triangle}u_h, \overline{\triangle}u_h) - \nu(\overline{\triangle}u_h, \nabla p_h) - \nu(\nabla p_h, \overline{\triangle}u_h) \\
&+ \nu(\overline{\triangle}u_h, f) + \nu(f, \overline{\triangle}u_h) - (\nabla p_h, f) \\
&- (f, \nabla p_h) + (\nabla p_h, \nabla p_h) + |f|_0^2.
\end{aligned} \tag{8.4}$$

ここで Lemma 4.5 の証明より

$$(\overline{\triangle}u_h, \overline{\triangle}u_h) = \boldsymbol{f}^T G_a^T E_1 G_a \boldsymbol{f}, \tag{8.5}$$

$$(\overline{\triangle}u_h, \nabla p_h) = (\nabla p_h, \overline{\triangle}u_h) = \boldsymbol{f}^T G_a E_2 \boldsymbol{f}, \tag{8.6}$$

$$(\nabla p_h, \nabla p_h) = \boldsymbol{f}^T G_b^T \tilde{D} G_b \boldsymbol{f} \tag{8.7}$$

である。また、

$$\begin{aligned}
|f|_0^2 &= (f_1, f_1) + (f_2, f_2) \\
&= \sum_{i=1}^{\hat{n}} f_1(P_i) \sum_{j=1}^{\hat{n}} (\hat{\phi}_i, \hat{\phi}_j) f_1(P_j) + \sum_{i=1}^{\hat{n}} f_2(P_i) \sum_{j=1}^{\hat{n}} (\hat{\phi}_i, \hat{\phi}_j) f_2(P_j) \\
&= \tilde{\boldsymbol{f}}_1^T \hat{L} \tilde{\boldsymbol{f}}_1 + \tilde{\boldsymbol{f}}_2^T \hat{L} \tilde{\boldsymbol{f}}_2
\end{aligned} \tag{8.8}$$

である。次に

$$\begin{aligned}
\hat{E}_x &= \left(\left(\frac{\partial \psi_i}{\partial x}, \hat{\phi}_j \right) \right), \\
\hat{E}_y &= \left(\left(\frac{\partial \psi_i}{\partial y}, \hat{\phi}_j \right) \right), \\
\hat{E} &= \hat{E}_x \tilde{\boldsymbol{f}}_1 + \hat{E}_y \tilde{\boldsymbol{f}}_2
\end{aligned}$$

とする。すると、

$$\begin{aligned}
(\nabla p_h, f) &= \sum_{i=1}^{m} b_i (\nabla \psi_i, f) \\
&= \sum_{i=1}^{m} b_i \left(\frac{\partial \psi_i}{\partial x}, f_1 \right) + \sum_{i=1}^{m} b_i \left(\frac{\partial \psi_i}{\partial y}, f_2 \right) \\
&= \sum_{i=1}^{m} b_i \sum_{j=1}^{\hat{n}} \left(\frac{\partial \psi_i}{\partial x}, \hat{\phi}_j \right) f_1(P_j) + \sum_{i=1}^{m} b_i \sum_{j=1}^{\hat{n}} \left(\frac{\partial \psi_i}{\partial y}, \hat{\phi}_j \right) f_2(P_j) \\
&= \boldsymbol{b} \hat{E}_x \tilde{\boldsymbol{f}}_1 + \boldsymbol{b} \hat{E}_y \tilde{\boldsymbol{f}}_2 \\
&= \boldsymbol{b} \hat{E} \\
&= \boldsymbol{f}^T G_b^T \hat{E} \qquad (8.9)
\end{aligned}$$

を得る。もちろん

$$(f, \nabla p_h) = (\nabla p_h, f) \qquad (8.10)$$

である。さらに

$$\begin{aligned}
\hat{K}^x &= \left(\left(\frac{\partial \hat{\phi}_i}{\partial x}, \hat{\phi}_j \right) \right), \\
\hat{K}^y &= \left(\left(\frac{\partial \hat{\phi}_i}{\partial y}, \hat{\phi}_j \right) \right), \\
E_4 &= \begin{pmatrix} M^x \hat{K}^x + M^y \hat{K}^y & O \\ O & M^x \hat{K}^x + M^y \hat{K}^y \end{pmatrix}, \\
\tilde{\boldsymbol{f}} &= (\tilde{\boldsymbol{f}}_1 \ \tilde{\boldsymbol{f}}_2)^T
\end{aligned}$$

とする。すると、

$$\begin{aligned}
(\overline{\triangle} u_h, f) &= (\operatorname{div} \overline{\nabla} u_h^{(1)}, f_1) + (\operatorname{div} \overline{\nabla} u_h^{(2)}, f_2) \\
&= \sum_{i=1}^{\hat{n}} c_i^{(1)} \sum_{j=1}^{\hat{n}} \left(\frac{\partial \hat{\phi}_i}{\partial x}, \hat{\phi}_j \right) \hat{f}_1(P_j) + \sum_{i=1}^{\hat{n}} d_i^{(1)} \sum_{j=1}^{\hat{n}} \left(\frac{\partial \hat{\phi}_i}{\partial y}, \hat{\phi}_j \right) \hat{f}_1(P_j) \\
&+ \sum_{i=1}^{\hat{n}} c_i^{(2)} \sum_{j=1}^{\hat{n}} \left(\frac{\partial \hat{\phi}_i}{\partial x}, \hat{\phi}_j \right) \hat{f}_2(P_j) + \sum_{i=1}^{\hat{n}} d_i^{(2)} \sum_{j=1}^{\hat{n}} \left(\frac{\partial \hat{\phi}_i}{\partial y}, \hat{\phi}_j \right) \hat{f}_2(P_j) \\
&= \boldsymbol{c}_1 \hat{K}^x \tilde{\boldsymbol{f}}_1 + \boldsymbol{d}_1 \hat{K}^y \tilde{\boldsymbol{f}}_1 + \boldsymbol{c}_2 \hat{K}^x \tilde{\boldsymbol{f}}_2 + \boldsymbol{d}_2 \hat{K}^y \tilde{\boldsymbol{f}}_2 \\
&= \boldsymbol{a}_1 M^x \hat{K}^x \tilde{\boldsymbol{f}}_1 + \boldsymbol{a}_1 M^y \hat{K}^y \tilde{\boldsymbol{f}}_1 + \boldsymbol{a}_2 M^x \hat{K}^x \tilde{\boldsymbol{f}}_2 + \boldsymbol{a}_2 M^y \hat{K}^y \tilde{\boldsymbol{f}}_2 \\
&= (\boldsymbol{a}_1 \ \boldsymbol{a}_2) \begin{pmatrix} M^x \hat{K}^x + M^y \hat{K}^y & O \\ O & M^x \hat{K}^x + M^y \hat{K}^y \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{f}}_1 \\ \tilde{\boldsymbol{f}}_2 \end{pmatrix} \\
&= \boldsymbol{f}^T G_a E_4 \tilde{\boldsymbol{f}} \qquad (8.11)
\end{aligned}$$

を得る。もちろん

$$(f, \overline{\triangle} u_h) = (\overline{\triangle} u_h, f) \qquad (8.12)$$

である。(8.4)-(8.12) より

$$
\begin{aligned}
|\nu\overline{\triangle}u_h - \nabla p_h + f|_0^2 &= \nu^2(\boldsymbol{f}^T G_a^T E_1 G_a \boldsymbol{f}) - 2\nu(\boldsymbol{f}^T G_a E_2 \boldsymbol{f}) + 2\nu(\boldsymbol{f}^T G_a E_4 \tilde{\boldsymbol{f}}) \\
&- 2(\boldsymbol{f}^T G_b^T \hat{E}) + (\boldsymbol{f}^T G_b^T \tilde{D} G_b \boldsymbol{f}) + (\tilde{\boldsymbol{f}}_1 \hat{L} \tilde{\boldsymbol{f}}_1 + \tilde{\boldsymbol{f}}_2 \hat{L} \tilde{\boldsymbol{f}}_2) \qquad (8.13)
\end{aligned}
$$

を得る。(8.2), (8.3), (8.13) から (8.1) の $C(u_h, p_h)$ の計算が可能になる。

行列 $\hat{K^x}$, $\hat{K^y}$ は前節の要素係数行列 $K_x^{(0)}$, $K_y^{(0)}$ から作成可能である。行列 $\hat{E}_x$, $\hat{E}_y$ の要素係数行列 $\hat{E}_x^{(0)}$, $\hat{E}_y^{(0)}$ を Mathematica を用いて求めた (プログラムは 9.2.1 節参照)。以下に $\hat{E}_x^{(0)}$, $\hat{E}_y^{(0)}$ を示す。

$$
\hat{E}_x^{(0)} = \frac{h_y}{36}\begin{pmatrix}
-1 & -1 & 0 & 0 & -4 & -2 & 0 & -2 & -8 \\
1 & 1 & 0 & 0 & 4 & 2 & 0 & 2 & 8 \\
0 & 0 & 1 & 1 & 0 & 2 & 4 & 2 & 8 \\
0 & 0 & -1 & -1 & 0 & -2 & -4 & -2 & -8
\end{pmatrix},
$$

$$
\hat{E}_y^{(0)} = \frac{h_x}{36}\begin{pmatrix}
-1 & 0 & 0 & -1 & -2 & 0 & -2 & -4 & -8 \\
0 & -1 & -1 & 0 & -2 & -4 & -2 & 0 & -8 \\
0 & 1 & 1 & 0 & 2 & 4 & 2 & 0 & 8 \\
1 & 0 & 0 & 1 & 2 & 0 & 2 & 4 & 8
\end{pmatrix}.
$$

8.2 実験結果

9.3 節のプログラムを用いて事後誤差評価式に表れる定数

$$
C(u_h, p_h) = \nu|\overline{\nabla}u_h - \nabla u_h|_0 + C_0 h|\nu\overline{\triangle}u_h - \nabla p_h + f|_0 + |\operatorname{div} u_h|_0
$$

を求める実験を行った。プログラムは前節と同様 MATLAB で行い、区間演算には INTLAB を用いた。$f = [f_1, f_2]^T$ は

$$
\begin{aligned}
f_1 &= 50(-2x + y + xy), \\
f_2 &= 20(1 - 5xy),
\end{aligned}
$$

とした。$\Omega = (0,1)\times(0,1)$, $\nu = 1$, $N = N_x = N_y$ で行った。$C_0 = 1/2\pi$ である。図 4 は $N = 15$ で外力密度 f と有限要素解 u_h, p_h をプロットした図である。以下のような結果になった。

N	アルゴリズム	$C(u_h, p_h)$
5	floating	4.980313575682848e-001
	interval	4.980313577728268e-001
10	floating	1.229866804907601e-001
	interval	1.229866892062705e-001
15	floating	5.427010506475726e-002
	interval	5.427020490506425e-002

$|u - u_h|_1$ と $|p - p_h|_0$ に関する事後誤差限界

$$
\left(\frac{1}{\nu^2} + \frac{1}{\beta^2}\right)^{\frac{1}{2}} C(u_h, p_h), \quad \left(\frac{1}{\beta} + \frac{\nu}{\beta^2}\right) C(u_h, p_h)
$$

は以下のようになった。

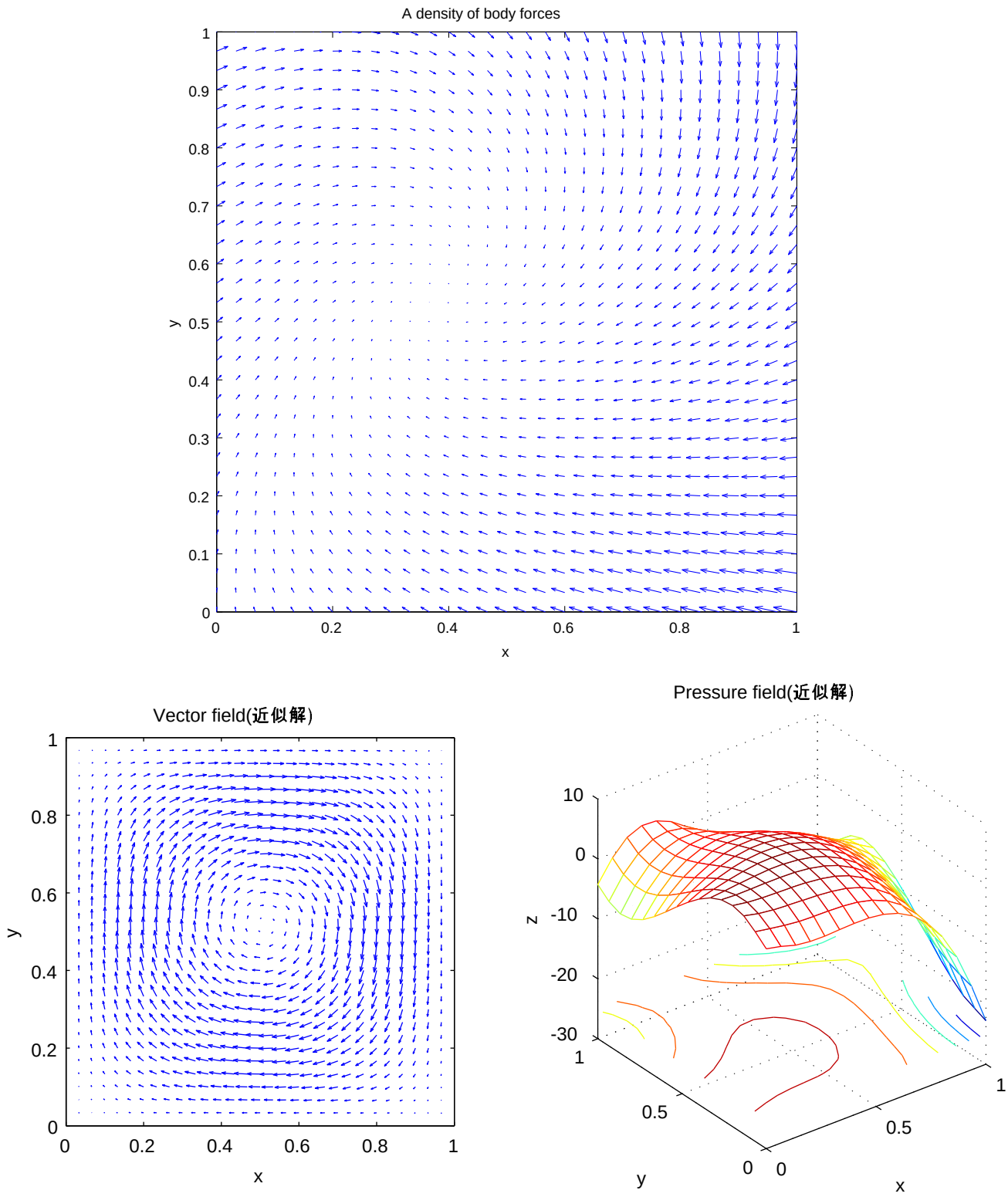


図 4: $N = 15$ での外力密度 f と有限要素解 u_h, p_h

N	アルゴリズム	$\lvert u-u_h\rvert_1$	$\lvert p-p_h\rvert_0$
5	floating	1.393458197026232e+000	4.702189485285095e+000
	interval	1.393458197598529e+000	4.702189487216293e+000
10	floating	3.441084490977951e-001	1.161185268850195e+000
	interval	3.441084734832141e-001	1.161185351138176e+000
15	floating	1.518440989844544e-001	5.123940762421375e-001
	interval	1.518443783309094e-001	5.123950188896931e-001

ここで、$\dfrac{1}{\beta}=\sqrt{4+2\sqrt{2}}$ である。floating で $N=2$ から $N=15$ まで求めた結果を対数グラフで図 5 に示す。傾きを調べると分割数のおよそ 2.02 乗に比例して値が減少していくことが

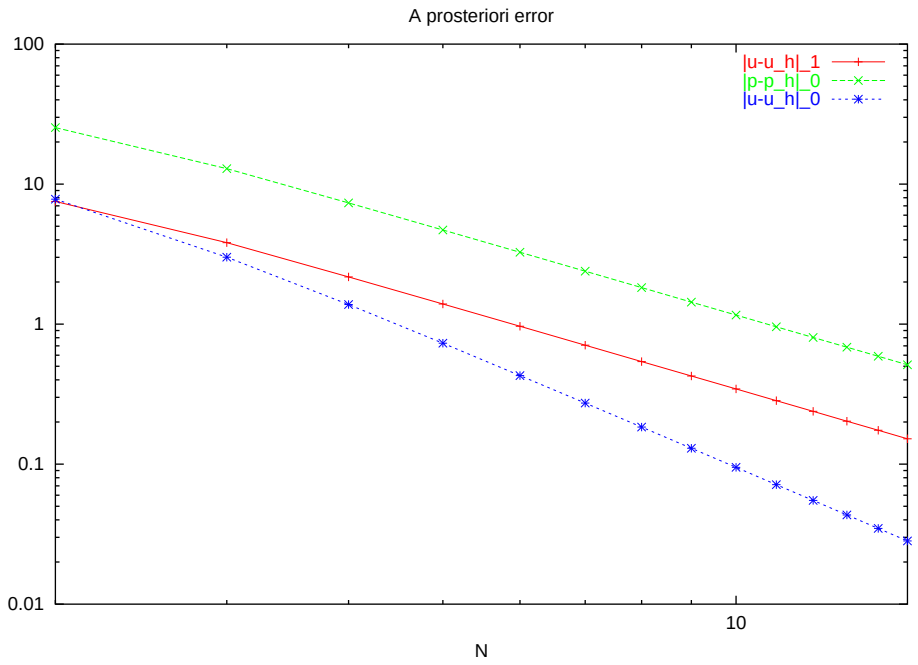


図 5: 事後誤差限界

分かる。

次に、$\lvert u-u_h\rvert_0$ に関する事後誤差評価式

$$\lvert u-u_h\rvert_0 \le \nu C_2^{(1)}\lvert u-u_h\rvert_1 + C_2^{(2)}\lvert \operatorname{div} u_h\rvert_0 + K_3\lvert p-p_h\rvert_0$$

の右辺を評価した結果を以下に示す。ここで $C_2^{(1)}$, $C_2^{(2)}$, K_3 は

$$\begin{aligned} C_2^{(1)} &= \left(\frac{1}{\nu^2}+\frac{1}{\beta^2}\right)^{\frac{1}{2}} C_2(h), \\ C_2^{(2)} &= \left(\frac{1}{\beta}+\frac{\nu}{\beta^2}\right) C_2(h), \\ \lvert \operatorname{div} u_h\rvert_0 &\le K_3\lvert P_0 f\rvert_0 \end{aligned}$$

で定まる値であり、前節ですでに求まっている。interval では $C_2^{(1)}$, $C_2^{(2)}$, K_3 の値に改良近似対角化法での値を用いた。

N	アルゴリズム	$\|u-u_h\|_0$
5	floating	7.309813930299081e-001
	interval	7.309813935469265e-001
10	floating	9.485496392558747e-002
	interval	9.485497384754534e-002
15	floating	2.826032200422597e-002
	interval	2.826039100769279e-002

floating で $N=2$ から $N=15$ まで求めた結果を対数グラフで図 5 に示す。傾きを調べると分割数のおよそ 3 乗に比例して値が減少していくことが分かる。

9 プログラムリスト

ここで紹介しているプログラムは 9.2.1 節のプログラムを除いてすべて MATLAB のプログラムである。また、MATLAB 上で区間演算を実現するツールボックスである S.M.Rump による INTLAB (`http://www.ti3.tu-harburg.de/~rump/intlab/`) を用いている。

9.1 一般化固有値の絶対値最大を求めるプログラム

ADM.m

```
% ADM (近似対角化法)
% 一般化固有値問題 Ax=λ Bx の固有値λの絶対値最大を精度保証付きで求める
% 区間行列 A, B に対して ADM(A,B) と用いる

function ret = ADM(intA,intB)

% Step 1. Cholesky 分解 (interval)
intC=intchol(intB);

% Step 2. E=inv(C)*A*inv(C)' を計算 (interval)
intE=inv(intC)*intA*inv(intC)';
E=mid(intE);
E=1/2*(E+E');

% Step 3. E の正規化された固有ベクトルにより近似対角化行列 T を作成 (floating)
[T D]=eig(E);

% Step 4. D=T'*E*T, lambda1 を計算 (interval)
intD=T'*intE*T;
lambda1=norm(intD,inf);

% Step 5. I=inv(T*T'), lambda2 を計算 (interval)
intT=intval(T);
I=inv(intT*intT');
lambda2=norm(I,inf);

% Step 6. lambda1*lambda2 を計算 (interval)
ret=intval(lambda1)*intval(lambda2);
```

RUMP.m

```
% RUMP (RUMP の方法)
% 一般化固有値問題 Ax=λ Bx の固有値λの絶対値最大を精度保証付きで求める
% 区間行列 A, B に対して RUMP(δ , δの増分,A,B) と用いる

function ret = ADM(delta,ddelta,intA,intB)

[n,n]=size(intA);
count=0;

% Step 1. B を C C' と Cholesky 分解 (interval)
intC=intchol(intB);

% Step 2. E = inv(C) * A * inv(C)' を計算 (interval)
intE=inv(intC)*intA*inv(intC)';
E=mid(intE);
E=1/2*(E+E');

% Step 3. E の 固有値の絶対値最大 beta_tilde を近似計算 (floating)
opts.disp=0;
beta_tilde=abs(eigs(E,1,'lm',opts));

% Step 4. ある微小な数 0 < delta << 1 により beta = (1 + delta)*beta_tilde とする (interval)
while true
    beta=(intval(1)+intval(delta))*intval(beta_tilde);

    % Step 5. X_1 = -E + beta * I_n , X_2 = E + beta * I_n を計算 (interval)
    intX_1=-intE+beta*eye(n);
    intX_2= intE+beta*eye(n);
    X_1=mid(intX_1);
    X_2=mid(intX_2);

    % Step 6. X_k (k=1,2) を Cholesky 分解 (X_k ≒ C_k_tilde * C_k_tilde') する (floating)
    %         Cholesky 分解が失敗する場合は delta の値を大きくして Step 4 に戻るか停止
    [C_1_tilde p1]=chol(X_1);
    [C_2_tilde p2]=chol(X_2);
    if p1==0 && p2==0
        break;
    end
    % delta を増やす
    delta=delta+ddelta;

    count=count+1;
    if delta > 0.5
        fprintf('停止します。%d 回δを大きくしました。\n δ=%f です。\n',count,delta);
        ret=0;
        return
    end
end
C_1_tilde=C_1_tilde';
C_2_tilde=C_2_tilde';

% Step 7. Y_k = C_k_tilde * C_k_tilde' - X_k, lambda_k = norm(Y_k,inf) (k=1,2)
%         を計算 (interval)
Y_1=intval(C_1_tilde)*intval(C_1_tilde')-intX_1;
Y_2=intval(C_2_tilde)*intval(C_2_tilde')-intX_2;
lambda1=norm(Y_1,inf);
lambda2=norm(Y_2,inf);

```

```
59  % Step 8. beta + max(lambda1,lambda2) を計算 (interval)
60  ret=beta+intval(lambda1)*intval(lambda2);
```

次のプログラムは ADM.m と RUMP.m から呼ばれる関数のプログラムである。

intchol.m

```
 1  % intchol.m
 2  % 区間行列を Cholesky 分解する
 3  % 区間行列 A に対して A=CC' となる行列 C を計算する
 4  % intchol(A) と用いる
 5  %
 6  % 計算が遅くあまり使い物にならない
 7
 8  function ret = intchol(A)
 9    n=size(A);
10    n=n(1,1);
11    L=intval(zeros(n));
12    for i=1:n
13        for j=1:i-1
14            s=A(i,j);
15            for k=1:j-1
16                s=s-L(i,k)*L(j,k);
17            end
18            L(i,j)=s/L(j,j);
19        end
20        s=A(i,i);
21        for k=1:i-1
22            s=s-L(i,k)^2;
23        end
24        L(i,i)=sqrt(s);
25    end
26    ret=L;
```

ADM_a.m

```
 1  % ADM_a (改良型近似対角化法)
 2  % 一般化固有値問題 Ax=λ Bx の固有値λの絶対値最大を精度保証付きで求める
 3  % 区間行列 A, B に対して ADM_a(A,B) と用いる
 4
 5  function ret = ADM_a(intA,intB)
 6
 7  [n,n]=size(intA);
 8
 9  A=mid(intA);
10  B=mid(intB);
11
12  % Step 1. Cholesky 分解 B ≒ C_tilde * C_tilde' を行う (floating)
13  C_tilde=chol(B)';
14
15  % Step 2. E_tilde = inv(C_tilde) * A * inv(C_tilde)' を計算 (floating)
16  E_tilde=inv(C_tilde)*A*inv(C_tilde)';
17  E_tilde=1/2*(E_tilde+E_tilde');
18
19  % Step 3. E_tilde の正規化された固有ベクトルにより近似対角化行列 T_tilde を作成 (floating)
20  [T_tilde D]=eig(E_tilde);
21
22  % Step 4. P_tilde = T_tilde' * inv(C_tilde) を計算 (floating)
```

```
23  P_tilde=T_tilde'*inv(C_tilde);
24
25  % Step 5. D = P_tilde * A * P_tilde', lambda1 = norm(D,inf) を計算 (interval)
26  intD=P_tilde*intA*P_tilde';
27  lambda1=norm(intD,inf);
28
29  % Step 6. I = inv(P_tilde * B * P_tilde'), lambda2 = norm(I,inf) を計算 (interval)
30  I=inv(P_tilde*intB*P_tilde');
31  lambda2=norm(I,inf);
32
33  % Step 7. lambda1 * lambda2 を計算 (interval)
34  ret=intval(lambda1)*intval(lambda2);
```

次のプログラムは G_RUMP.m から呼ばれる関数のプログラムである。

VPD.m

```
 1  % VPD (正定値性判定法)
 2  % 実対称行列の正定値性を判定する
 3  % 正定値性を判定する区間行列 X に対し VPD(δ, δの増分,X) と用いる
 4  % X が正定値と判定できた場合は 1 を X が正定値と判定できなかった場合は 0 を返す。
 5  %
 6  % delta = 0.5 で停止としている
 7
 8  function ret = VPD(delta,ddelta,intX)
 9
10  if delta >= 0.5 || delta <=0
11      fprintf('0 < δ << 1 として下さい\n');
12      ret=0;
13      return
14  end
15
16  X=mid(intX);
17  X=1/2*(X+X');
18  [n,n]=size(X);
19  count=0;
20
21  % Step 1. X の最小固有値 rho_tilde を求める (floating)
22  %         rho_tilde ≦ 0 の場合停止
23  rho_tilde=min(eig(X));
24  % opts.disp=0;
25  % rho_tilde=eigs(X,1,'sa',opts);
26  if rho_tilde<=0
27      ret=0;
28      return
29  end
30
31  % Step 2. ある微小な数 0 < delta << 1 により rho = (1 - delta) * rho_tilde とする (interval)
32  while true
33      while true
34          rho=(intval(1)-intval(delta))*intval(rho_tilde);
35
36          % Step 3. Y = X - rho * I_n を計算 (interval)
37          intY=intX-rho*eye(n);
38          Y=mid(intY);
39
40          % Step 4. Y を Cholesky 分解 (Y ≒ C_tilde * C_tilde') する (floating)
41          %         Cholesky 分解が失敗する場合は delta の値を大きくして Step 2 に戻るか停止
```

```
42          [C_tilde,p]=chol(Y);
43          if p==0
44              break;
45          end
46          % delta を増やす
47          delta=delta+ddelta;
48
49          count=count+1;
50          if delta > 0.5
51              %fprintf('停止します。%d回δを大きくしました。\n δ=%f です。\n',count,delta);
52              ret=0;
53              return
54          end
55      end
56      C_tilde=C_tilde';
57
58      % Step 5. Z = C_tilde * C_tilde' - Y, lambda = norm(Z,inf) を計算 (interval)
59      C_tilde=intval(C_tilde);
60      Z=C_tilde*C_tilde'-intY;
61      lambda=norm(Z,inf);
62
63      % Step 6. rho - lambda > 0 ならば終了 そうでない場合は delta の値を大きくして Step 2 に
戻るか停止
64      if inf(rho)-lambda > 0
65          ret=1;
66          return
67      end
68      % delta を増やす
69      delta=delta+ddelta;
70
71      count=count+1;
72      if delta > 0.5
73          %fprintf('停止します。%d回δを大きくしました。\n δ=%f です。\n',count,delta);
74          ret=0;
75          return
76      end
77  end
```

G_RUMP.m

```
 1  % G_RUMP (一般化 RUMP 法)
 2  % 一般化固有値問題 Ax=λ Bx の固有値λの絶対値最大を精度保証付きで求める
 3  % 区間行列 A, B に対して G_RUMP(δ, δの増分,VPDのδ,VPDのδの増分,A,B) と用いる
 4
 5  function ret = G_RUMP(delta,ddelta,VPDdelta,VPDddelta,intA,intB)
 6
 7  A=mid(intA);
 8  B=mid(intB);
 9  A=1/2*(A+A');
10  B=1/2*(B+B');
11
12  count=0;
13
14  % Step 1. γの近似値 beta_tilde を計算 (floating)
15  opts.disp=0;
16  beta_tilde=abs(eigs(A,B,1,'lm',opts));
17  %beta_tilde=max(abs(eig(A,B)));
```

```
18
19  % Step 2. ある微小な数 0 < delta << 1 により beta = (1 + delta) * beta_tilde とする (interval)
20  while true
21      beta=(intval(1)+intval(delta))*intval(beta_tilde);
22
23      % Step 3. X_1 = -A + beta * B, X_2 = A + beta * B を計算 (interval)
24      X_1=-intA+beta*intB;
25      X_2= intA+beta*intB;
26
27      % Step 4. VPD により X_1, X_2 の正定値性を判定. ともに正ならば終了. 正定値性が得られなかっ
た場合は
28      %         delta の値を増やして Step 2 に戻るか停止
29      if VPD(VPDdelta,VPDddelta,X_1)==1 & VPD(VPDdelta,VPDddelta,X_2)==1
30          ret=beta;
31          return
32      end
33      delta=delta+ddelta;
34      count=count+1;
35      if delta > 0.5
36          fprintf('停止します。%d 回δを大きくしました。\n δ=%f です。\n',count,delta);
37          ret=0;
38          return
39      end
40  end
```

9.2 アプリオリ誤差評価の数値実験で用いたプログラム

9.2.1 要素係数行列を求めるプログラム

以下は、7.2 節と 8.1 節の要素係数行列を求める Mathematica プログラムである。

make-matrix-mathematica.m

```
 1  L0[x_,h_]:=1-x/h
 2  L1[x_,h_]:=x/h
 3  varphi[i_,x_,h_]:=Part[{L0[x,h](2L0[x,h]-1),4L0[x,h]L1[x,h],L1[x,h](2L1[x,h]-1)},i+1]
 4  phi2[i_,j_,x_,y_]:=varphi[i,x,hx]varphi[j,y,hy]
 5  pusai1[i_,x_,h_]:=Part[{L0[x,h], L1[x,h]},i+1]
 6  pusai2[i_,j_,x_,y_]:=pusai1[i,x,hx]pusai1[j,y,hy]
 7  mylist={{0,0},{2,0},{2,2},{0,2},{1,0},{2,1},{1,2},{0,1},{1,1}};
 8  mylist2={{0,0},{1,0},{1,1},{0,1}};
 9  phi[n_,x_,y_]:=phi2[mylist[[n]][[1]],mylist[[n]][[2]],x,y]
10  pusai[n_,x_,y_]:=pusai2[mylist2[[n]][[1]],mylist2[[n]][[2]],x,y]
11
12  Lhat0=900/(hx*hy) Table[Integrate[phi[i,x,y]phi[j,x,y],{x,0,hx},{y,0,hy}],
13      {i,9},{j,9}];
14  Dhatxx0=90*hx/hy Table[Integrate[D[phi[i,x,y],x]*D[phi[j,x,y],x],{x,0,hx},{y,0,hy}],
15      {i,9},{j,9}];
16  Dhatxy0=36*Table[Integrate[D[phi[i,x,y],x]*D[phi[j,x,y],y],{x,0,hx},{y,0,hy}],
17      {i,9},{j,9}];
18  Dhatyy0=90*hy/hx Table[Integrate[D[phi[i,x,y],y]*D[phi[j,x,y],y],{x,0,hx},{y,0,hy}],
19      {i,9},{j,9}];
20  Kx0=180/hy*Table[Integrate[D[phi[i,x,y],x]*phi[j,x,y],{x,0,hx},{y,0,hy}],
21      {i,9},{j,9}];
22  Ky0=180/hx*Table[Integrate[D[phi[i,x,y],y]*phi[j,x,y],{x,0,hx},{y,0,hy}],
23      {i,9},{j,9}];
```

```
Fhatxx0=6*hx/hy Table[Integrate[D[phi[i,x,y],x]*D[pusai[j,x,y],x],{x,0,hx},{y,0,hy}],
    {i,9},{j,4}];
Fhatxy0= 36*Table[Integrate[D[phi[i,x,y],x]*D[pusai[j,x,y],y],{x,0,hx},{y,0,hy}],
    {i,9},{j,4}];
Fhatyx0= 36*Table[Integrate[D[phi[i,x,y],y]*D[pusai[j,x,y],x],{x,0,hx},{y,0,hy}],
    {i,9},{j,4}];
Fhatyy0= 6*hy/hx Table[Integrate[D[phi[i,x,y],y]*D[pusai[j,x,y],y],{x,0,hx},{y,0,hy}],
    {i,9},{j,4}];
Dtilde0= 6*hx*hy Table[Integrate[D[pusai[i,x,y],x]*D[pusai[j,x,y],x],{x,0,hx},{y,0,hy}]
              +Integrate[D[pusai[i,x,y],y]*D[pusai[j,x,y],y],{x,0,hx},{y,0,hy}],{i,4},{j,4}];
Dtilde0=Expand[Dtilde0];
D00=90*hx*hy Table[Integrate[D[phi[i,x,y],x]*D[phi[j,x,y],x],{x,0,hx},{y,0,hy}]
              +Integrate[D[phi[i,x,y],y]*D[phi[j,x,y],y],{x,0,hx},{y,0,hy}],{i,9},{j,9}];
D00=Expand[D00];
Ex0= 36/hy Table[Integrate[pusai[i,x,y]*D[phi[j,x,y],x],{x,0,hx},{y,0,hy}],
    {i,4},{j,9}];
Ey0= 36/hx Table[Integrate[pusai[i,x,y]*D[phi[j,x,y],y],{x,0,hx},{y,0,hy}],
    {i,4},{j,9}];
Ehatx0=36/hy Table[Integrate[D[pusai[i,x,y],x]*phi[j,x,y],{x,0,hx},{y,0,hy}],
    {i,4},{j,9}];
Ehaty0=36/hx Table[Integrate[D[pusai[i,x,y],y]*phi[j,x,y],{x,0,hx},{y,0,hy}],
    {i,4},{j,9}];

Print["Lhat0="]
Lhat0
Print["D00="]
D00
Print["Dhatxx0="]
Dhatxx0
Print["Dhatxy0="]
Dhatxy0
Print["Dhatyy0="]
Dhatyy0
Print["Kx0="]
Kx0
Print["Ky0="]
Ky0
Print["Fhatxx0="]
Fhatxx0
Print["Fhatxy0="]
Fhatxy0
Print["Fhatyx0="]
Fhatyx0
Print["Fhatyy0="]
Fhatyy0
Print["Dtilde0="]
Dtilde0
Print["Ex0="]
Ex0
Print["Ey0="]
Ey0
Print["Ehatx0="]
Ehatx0
Print["Ehaty0="]
Ehaty0
```

9.2.2 floating で行列 $F,\ A_1,\ A_2,\ A_3,\ A_4$ を求めるプログラム

以下は floating で行列 $F,\ A_1,\ A_2,\ A_3,\ A_4$ を求めるプログラムである。

apriori_floating.m

```
% apriori_floating.m
% nakao,yamamoto,watanabe 論文での行列
% F, A1, A2, A3, A4 を floating で作成する

function [F,A1,A2,A3,A4]=apriori_floating(Nx,Ny,nu,width,height);

hx=width/Nx;
hy=height/Ny;

hx2=hx^2;
hy2=hy^2;

dim_phihat=(2*Nx+1)*(2*Ny+1);
dim_phi=(2*Nx-1)*(2*Ny-1);
dim_psi=(Nx+1)*(Ny+1)-1;

% beta の設定
nnode_y1=Ny+1;
beta=4*ones(dim_psi+1,1);
for i=2:2:2*Nx-2
    beta(i*nnode_y1/2+1)=2;
    beta(i*nnode_y1/2+Ny+1)=2;
end
for j=2:2:2*Ny-2
    beta(j/2+1)=2;
    beta(Nx*nnode_y1+j/2+1)=2;
end
beta(1)=1;
beta(Ny+1)=1;
beta(Nx*nnode_y1+1)=1;
beta(Nx*nnode_y1+Ny+1)=1;

% 行列 Lhat0 の作成
Lhat0=hx*hy/900*[16 -4  1 -4   8  -2  -2   8   4;
                 -4 16 -4  1   8   8  -2  -2   4;
                  1 -4 16 -4  -2   8   8  -2   4;
                 -4  1 -4 16  -2  -2   8   8   4;
                  8  8 -2 -2  64   4 -16   4  32;
                 -2  8  8 -2   4  64   4 -16  32;
                 -2 -2  8  8 -16   4  64   4  32;
                  8 -2 -2  8   4 -16   4  64  32;
                  4  4  4  4  32  32  32  32 256];
% 行列 Lhat の作成
Lhat=assem_floating1(Lhat0,Nx,Ny);

% 行列 L の作成
L=delete_boundary(Lhat,Nx,Ny);

% 行列 D00 の作成
D00=hx/(90*hy)*[28  -7  -1   4   14   8    2 -32  -16;
                -7  28   4  -1   14 -32    2   8  -16;
                -1   4  28  -7    2 -32   14   8  -16;
```

```
                  4  -1  -7  28    2   8   14 -32  -16;
                 14  14   2   2  112 -16   16 -16 -128;
                  8 -32 -32   8  -16  64  -16 -16   32;
                  2   2  14  14   16 -16  112 -16 -128;
                -32   8   8 -32  -16 -16  -16  64   32;
                -16 -16 -16 -16 -128  32 -128  32  256];
D00=D00+hy/(90*hx)*[28   4  -1  -7 -32    2   8   14  -16;
                     4  28  -7  -1 -32   14   8    2  -16;
                    -1  -7  28   4   8   14 -32    2  -16;
                    -7  -1   4  28   8    2 -32   14  -16;
                   -32 -32   8   8  64  -16 -16  -16   32;
                     2  14  14   2 -16  112 -16   16 -128;
                     8   8 -32 -32 -16  -16  64  -16   32;
                    14   2   2  14 -16   16 -16  112 -128;
                   -16 -16 -16 -16  32 -128  32 -128  256];

% 行列 D0 の作成
D0=delete_boundary(assem_floating1(D00,Nx,Ny),Nx,Ny);

% 行列 Dhatxx0 の作成
Dhatxx0=hy/(90*hx)*[28   4  -1  -7 -32    2   8   14  -16;
                     4  28  -7  -1 -32   14   8    2  -16;
                    -1  -7  28   4   8   14 -32    2  -16;
                    -7  -1   4  28   8    2 -32   14  -16;
                   -32 -32   8   8  64  -16 -16  -16   32;
                     2  14  14   2 -16  112 -16   16 -128;
                     8   8 -32 -32 -16  -16  64  -16   32;
                    14   2   2  14 -16   16 -16  112 -128;
                   -16 -16 -16 -16  32 -128  32 -128  256];

% 行列 Dhatxx の作成
Dhatxx=assem_floating1(Dhatxx0,Nx,Ny);

% 行列 Dhatxy0 の作成
Dhatxy0=1/36*[9  -3  -1   3  12   4   4 -12 -16;
              3  -9  -3   1 -12  12  -4  -4  16;
             -1   3   9  -3   4 -12  12   4 -16;
             -3   1   3  -9  -4  -4 -12  12  16;
            -12  12   4  -4   0 -16   0  16   0;
              4 -12  12  -4 -16   0  16   0   0;
              4  -4 -12  12   0  16   0 -16   0;
             12  -4   4 -12  16   0 -16   0   0;
            -16  16 -16  16   0   0   0   0   0];

% 行列 Dhatxy の作成
Dhatxy=assem_floating1(Dhatxy0,Nx,Ny);

% 行列 Dhatyy0 の作成
Dhatyy0=hx/(90*hy)*[28  -7  -1   4   14   8    2 -32  -16;
                    -7  28   4  -1   14 -32    2   8  -16;
                    -1   4  28  -7    2 -32   14   8  -16;
                     4  -1  -7  28    2   8   14 -32  -16;
                    14  14   2   2  112 -16   16 -16 -128;
                     8 -32 -32   8  -16  64  -16 -16   32;
                     2   2  14  14   16 -16  112 -16 -128;
                    -32   8   8 -32  -16 -16  -16  64   32;
                    -16 -16 -16 -16 -128  32 -128  32 256];

```

```
111 % 行列 Dhatyy の作成
112 Dhatyy=assem_floating1(Dhatyy0,Nx,Ny);
113
114 % 行列 Dxx の作成
115 Dxx=delete_boundary(Dhatxx,Nx,Ny);
116
117 % 行列 Dxy の作成
118 Dxy=delete_boundary(Dhatxy,Nx,Ny);
119
120 % 行列 Dyy の作成
121 Dyy=delete_boundary(Dhatyy,Nx,Ny);
122
123 % 行列 Kx0 の作成
124 Kx0=hy/180*[-12   4  -1   3 -16   2   4  -6  -8;
125              -4  12  -3   1  16   6  -4  -2   8;
126               1  -3  12  -4  -4   6  16  -2   8;
127               3  -1   4 -12   4   2 -16  -6  -8;
128              16 -16   4  -4   0  -8   0   8   0;
129              -2   6   6  -2   8  48   8 -16  64;
130              -4   4 -16  16   0  -8   0   8   0;
131              -6   2   2  -6  -8  16  -8 -48 -64;
132               8  -8  -8   8   0 -64   0  64   0];
133
134 % 行列 Kx の作成
135 Kx=assem_floating2(Kx0,Nx,Ny);
136
137 % 行列 Ky0 の作成
138 Ky0=hx/180*[-12   3  -1  4   -6   4   2 -16  -8;
139               3 -12   4  -1  -6 -16   2   4  -8;
140               1  -4  12  -3  -2  16   6  -4   8;
141              -4   1  -3  12  -2  -4   6  16   8;
142              -6  -6   2   2 -48  -8  16  -8 -64;
143              -4  16 -16   4   8   0  -8   0   0;
144              -2  -2   6   6 -16   8  48   8  64;
145              16  -4   4 -16   8   0  -8   0   0;
146               8   8  -8  -8  64   0 -64   0   0];
147
148 % 行列 Ky の作成
149 Ky=assem_floating2(Ky0,Nx,Ny);
150
151 % 行列 Fhatxx0 の作成
152 Fhatxx0=hy/(6*hx)*[ 1 -1  0  0;
153                    -1  1  0  0;
154                     0  0  1 -1;
155                     0  0 -1  1;
156                     0  0  0  0;
157                    -2  2  2 -2;
158                     0  0  0  0;
159                     2 -2 -2  2;
160                     0  0  0  0];
161
162 % 行列 Fhatxx の作成
163 Fhatxx=assem_floating3(Fhatxx0,beta,Nx,Ny);
164
165 % 行列 Fhatxy0 の作成
166 Fhatxy0=1/36*[  5   1  -1  -5;
167                -1  -5   5   1;
168                -1  -5   5   1;
```

```
169                  5   1  -1  -5;
170                 -4   4  -4   4;
171                 -4 -20  20   4;
172                 -4   4  -4   4;
173                 20   4  -4 -20;
174                -16  16 -16  16];
175
176  % 行列 Fhatxy の作成
177  Fhatxy=assem_floating3(Fhatxy0,beta,Nx,Ny);
178
179  % 行列 Fhatyx0 の作成
180  Fhatyx0=1/36*[  5  -5  -1   1;
181                  5  -5  -1   1;
182                 -1   1   5  -5;
183                 -1   1   5  -5;
184                 20 -20  -4   4;
185                 -4   4  -4   4;
186                 -4   4  20 -20;
187                 -4   4  -4   4;
188                -16  16 -16  16];
189
190  % 行列 Fhatyx の作成
191  Fhatyx=assem_floating3(Fhatyx0,beta,Nx,Ny);
192
193  % 行列 Fhatyy0 の作成
194  Fhatyy0=hx/(6*hy)*[ 1  0  0 -1;
195                      0  1 -1  0;
196                      0 -1  1  0;
197                     -1  0  0  1;
198                      2  2 -2 -2;
199                      0  0  0  0;
200                     -2 -2  2  2;
201                      0  0  0  0;
202                      0  0  0  0];
203
204  % 行列 Fhatyy の作成
205  Fhatyy=assem_floating3(Fhatyy0,beta,Nx,Ny);
206
207  % 行列 Dtilde0 の作成
208  Dtilde0=1/(6*hx*hy)*[2*hx2+2*hy2    hx2-2*hy2     -hx2-hy2   -2*hx2+hy2;
209                         hx2-2*hy2  2*hx2+2*hy2   -2*hx2+hy2     -hx2-hy2;
210                          -hx2-hy2   -2*hx2+hy2  2*hx2+2*hy2    hx2-2*hy2;
211                        -2*hx2+hy2     -hx2-hy2    hx2-2*hy2  2*hx2+2*hy2];
212
213  % 行列 Dtilde の作成
214  Dtilde=assem_floating4(Dtilde0,beta,Nx,Ny);
215
216  % 行列 Ex0 の作成
217  Ex0=hy/36*[-5 1 0  0  4  2  0 -10  8;
218             -1 5 0  0 -4 10  0  -2 -8;
219              0 0 5 -1  0 10 -4  -2 -8;
220              0 0 1 -5  0  2  4 -10  8];
221
222  % 行列 Ex の作成
223  Ex=assem_floating5(Ex0,beta,Nx,Ny);
224
225  % 行列 Ey0 の作成
226  Ey0=hx/36*[-5  0 0 1 -10  0  2  4  8;
```

```
227              0 -5 1 0 -10  4  2  0  8;
228              0 -1 5 0  -2 -4 10  0 -8;
229             -1  0 0 5  -2  0 10 -4 -8];
230
231 % 行列 Ey の作成
232 Ey=assem_floating5(Ey0,beta,Nx,Ny);
233
234 %                                   %
235 % 基底関数からの行列作成 ここまで %
236 %                                   %
237
238 % 行列 D の作成
239 D=[         nu*D0  zeros(dim_phi);
240    zeros(dim_phi)          nu*D0];
241
242 % 行列 E の作成
243 E=[Ex Ey];
244
245 % 行列 G の作成
246 G=[ D            -E';
247    -E zeros(dim_psi)];
248
249 % 行列 invG の作成
250 invG=inv(G);
251
252 % 行列 Ga, Gb, Gstar の作成
253 Ga=invG(1:2*dim_phi,1:2*dim_phi);
254 Gb=invG(2*dim_phi+1:2*dim_phi+dim_psi,1:2*dim_phi);
255 Gstar=invG(2*dim_phi+1:2*dim_phi+dim_psi,2*dim_phi+1:2*dim_phi+dim_psi);
256
257 % 行列 invLhat の作成
258 invLhat=inv(Lhat);
259
260 % 行列 Mx, My の作成
261 Mx=Kx*invLhat;
262 My=Ky*invLhat;
263
264 % 行列 invL の作成
265 invL=inv(L);
266
267 % 行列 F の作成
268 F=[         invL  zeros(dim_phi);
269    zeros(dim_phi)          invL];
270
271 % 行列 Q1 の作成
272 tmpQ1=D0-Mx*Kx'-My*Ky';
273 Q1=[         tmpQ1 zeros(dim_phi);
274     zeros(dim_phi)         tmpQ1];
275
276 % 行列 A1 の作成
277 A1=Ga*Q1*Ga;
278
279 % 行列 Ehatxx, Ehatxy, Ehatyy の作成
280 Ehatxx=Mx*Dhatxx*Mx';
281 Ehatxy=Mx*Dhatxy*My';
282 Ehatyy=My*Dhatyy*My';
283
284 % 行列 E1 の作成
```

```
285 tmpE1=Ehatxx+Ehatxy+Ehatxy'+Ehatyy;
286 E1=[         tmpE1 zeros(dim_phi);
287     zeros(dim_phi)         tmpE1];
288
289 % 行列 E2 の作成
290 E2=[(Mx*Fhatxx+My*Fhatyx)*Gb;
291     (Mx*Fhatxy+My*Fhatyy)*Gb];
292
293 % 行列 E3 の作成
294 tmpE3=-(Kx*invLhat*Kx'+Ky*invLhat*Ky')*invL;
295 E3=[        tmpE3  zeros(dim_phi);
296     zeros(dim_phi)         tmpE3];
297
298 % 行列 A2 の作成
299 A2=nu^2*Ga'*E1*Ga-nu*Ga*E2-nu*(Ga*E2)'+nu*Ga*E3+nu*(Ga*E3)'+Gb'*E*F+(Gb'*E*F)'+Gb'*Dtilde*Gb+F;
300
301 % 行列 Q3 の作成
302 Q3=[Dxx  Dxy;
303     Dxy' Dyy];
304
305 % 行列 A3 の作成
306 A3=Ga*Q3*Ga;
307
308 % 行列 A4 の作成
309 A4=Gb'*E*F+(Gb'*E*F)'+Gb'*Dtilde*Gb+F;
```

以下は apriori_floating.m から呼ばれる関数のプログラムである。

assem_floating1.m

```
 1 % assem_floating1.m
 2 % 要素係数行列から全体行列を作り出す (floating)
 3 % Lhat, Dxxhat, Dxyhat, Dyyhat に対応
 4
 5 function ret = assem_floating1(A0,Nx,Ny)
 6
 7 nnode_x=2*Nx+1;
 8 nnode_y=2*Ny+1;
 9
10 A=zeros(nnode_x*nnode_y);
11
12 num=[0 2*nnode_y 2*nnode_y+2 2 nnode_y 2*nnode_y+1 nnode_y+2 1 nnode_y+1];
13
14 for p=1:Nx
15     for q=1:Ny
16         i=2*p-2;
17         j=2*q-2;
18         % l : 要素 e_pq の局所節点番号 1 の全体節点番号
19         l=i*nnode_y+j+1;
20         for s=1:9
21             for t=1:9
22                 A(l+num(s),l+num(t))= A(l+num(s),l+num(t))+A0(s,t);
23             end
24         end
25     end
26 end
27 ret=A;
```

assem_floating2.m

```
% assem_floating2.m
% 要素係数行列から全体行列を作り出す (floating)
% Kx, Ky に対応

function ret = assem_floaring2(A0,Nx,Ny)

nnode_x=2*Nx+1;
nnode_y=2*Ny+1;

size_A=nnode_x*nnode_y;

A=zeros(size_A);

num=[0 2*nnode_y 2*nnode_y+2 2 nnode_y 2*nnode_y+1 nnode_y+2 1 nnode_y+1];

for p=1:Nx
    for q=1:Ny
        i=2*p-2;
        j=2*q-2;
        % l : 要素 e_pq の局所節点番号 1 の全体節点番号
        l=i*nnode_y+j+1;
        for s=1:9
            for t=1:9
                A(l+num(s),l+num(t))= A(l+num(s),l+num(t))+A0(s,t);
            end
        end
    end
end

count=1;
for n=nnode_y+2:size_A-nnode_y-1
    if mod(n,nnode_y)~=0 && mod(n,nnode_y)~=1
            nb_num(count)=n;
        count=count+1;
    end
end
A=A(nb_num,:);
ret=A;
```

assem_floating3.m

```
% assem_floating3.m
% 要素係数行列から全体行列を作り出す (floating)
% Fhatxx, Fhatxy, Fhatyx, Fhatyy に対応

function ret = assem_floating3(A0,beta,Nx,Ny)

nnode_x=2*Nx+1;
nnode_y=2*Ny+1;

nnode_x1=Nx+1;
nnode_y1=Ny+1;

A=zeros(nnode_x*nnode_y,nnode_x1*nnode_y1);

```

```
15  num=[0 2*nnode_y 2*nnode_y+2 2 nnode_y 2*nnode_y+1 nnode_y+2 1 nnode_y+1];
16  num1=[0 nnode_y1 nnode_y1+1 1];
17
18  for p=1:Nx
19      for q=1:Ny
20          i=2*p-2;
21          j=2*q-2;
22          % l : 要素 e_pq の局所節点番号 1 の Φ hat での全体節点番号
23          % m : 要素 e_pq の局所節点番号 1 の φ     での全体節点番号
24          l=i*nnode_y+j+1;
25          m=i*nnode_y1/2+j/2+1;
26          for s=1:9
27              for t=1:4
28                  A(l+num(s),m+num1(t))= A(l+num(s),m+num1(t))+A0(s,t);
29              end
30          end
31      end
32  end
33  for i=1:nnode_x*nnode_y
34      for j=1:nnode_x1*nnode_y1
35          A(i,j)=A(i,j)-beta(j)*A(i,nnode_x1*nnode_y1);
36      end
37  end
38  A(:,nnode_x1*nnode_y1)=[];
39  ret=A;
```

assem_floating4.m

```
 1  % assem_floating4.m
 2  % 要素係数行列から全体行列を作り出す (floating)
 3  % Dtilde に対応
 4
 5  function ret = assem_floating4(A0,beta,Nx,Ny)
 6
 7  nnode_x1=Nx+1;
 8  nnode_y1=Ny+1;
 9
10  A=zeros(nnode_x1*nnode_y1);
11
12  num1=[0 nnode_y1 nnode_y1+1 1];
13
14  for p=1:Nx
15      for q=1:Ny
16          i=2*p-2;
17          j=2*q-2;
18          % m : 要素 e_pq の局所節点番号 1 の φ での全体節点番号
19          m=i*nnode_y1/2+j/2+1;
20          for s=1:4
21              for t=1:4
22                  A(m+num1(s),m+num1(t))=A(m+num1(s),m+num1(t))+A0(s,t);
23              end
24          end
25      end
26  end
27
28  for i=1:nnode_x1*nnode_y1
29      for j=1:nnode_x1*nnode_y1
```

```
30          m=nnode_x1*nnode_y1;
31          A(i,j)=A(i,j)-beta(i)*A(m,j)-beta(j)*A(m,i)+beta(i)*beta(j)*A(m,m);
32      end
33  end
34
35  A(nnode_x1*nnode_y1,:)=[];
36  A(:,nnode_x1*nnode_y1)=[];
37  ret=A;
```

assem_floating5.m

```
 1  % assem_floating5.m
 2  % 要素係数行列から全体行列を作り出す (floating)
 3  % Ex, Ey に対応
 4
 5  function ret = assem_floating5(A0,beta,Nx,Ny)
 6
 7  nnode_x=2*Nx+1;
 8  nnode_y=2*Ny+1;
 9
10  size_A=nnode_x*nnode_y;
11
12  nnode_x1=Nx+1;
13  nnode_y1=Ny+1;
14
15  A=zeros(nnode_x1*nnode_y1,nnode_x*nnode_y);
16
17  num=[0 2*nnode_y 2*nnode_y+2 2 nnode_y 2*nnode_y+1 nnode_y+2 1 nnode_y+1];
18  num1=[0 nnode_y1 nnode_y1+1 1];
19
20  for p=1:Nx
21      for q=1:Ny
22          i=2*p-2;
23          j=2*q-2;
24          % l : 要素 e_pq の局所節点番号 1 の Φ hat での全体節点番号
25          % m : 要素 e_pq の局所節点番号 1 の φ     での全体節点番号
26          l=i*nnode_y+j+1;
27          m=i*nnode_y1/2+j/2+1;
28          for s=1:4
29              for t=1:9
30                  A(m+num1(s),l+num(t))= A(m+num1(s),l+num(t))+A0(s,t);
31              end
32          end
33      end
34  end
35
36  for i=1:nnode_x1*nnode_y1
37      for j=1:nnode_x*nnode_y
38          A(i,j)=A(i,j)-beta(i)*A(nnode_x1*nnode_y1,j);
39      end
40  end
41
42  count=1;
43  for n=nnode_y+2:size_A-nnode_y-1
44      if mod(n,nnode_y)~=0 && mod(n,nnode_y)~=1
45              nb_num(count)=n;
46          count=count+1;
```

```
47      end
48  end
49  A=A(:,nb_num);
50  A(nnode_x1*nnode_y1,:)=[];
51  ret=A;
```

delete_boundary.m

```
 1  % delete_boundary.m
 2  % 境界に属する節点に対応する行と列を削除する (floating,interval)
 3  % L, D0, Dxx, Dxy, Dyy に対応
 4
 5  function ret = delete_boundary(Ahat,Nx,Ny)
 6
 7  nnode_x=2*Nx+1;
 8  nnode_y=2*Ny+1;
 9
10  size_A=nnode_x*nnode_y;
11  A=zeros(size_A);
12
13  count=1;
14  for n=nnode_y+2:size_A-nnode_y-1
15      if mod(n,nnode_y)~=0 && mod(n,nnode_y)~=1
16              nb_num(count)=n;
17          count=count+1;
18      end
19  end
20  A=Ahat(nb_num,nb_num);
21  ret=A;
```

9.2.3 interval で行列 $F,\ A_1,\ A_2,\ A_3,\ A_4$ を求めるプログラム

以下は interval で行列 $F,\ A_1,\ A_2,\ A_3,\ A_4$ を求めるプログラムである。delete_boundary.m は floating のプログラムと同じものが利用できる。

apriori_interval.m

```
 1  % apriori_interval.m
 2  % nakao,yamamoto,watanabe 論文での行列
 3  % F, A1, A2, A3, A4 を interval で作成する
 4
 5  function [F,A1,A2,A3,A4]=apriori_interval(Nx,Ny,nu,width,height);
 6
 7  hx=intval(width)/Nx;
 8  hy=intval(height)/Ny;
 9
10  hx2=hx^2;
11  hy2=hy^2;
12
13  dim_phihat=(2*Nx+1)*(2*Ny+1);
14  dim_phi=(2*Nx-1)*(2*Ny-1);
15  dim_psi=(Nx+1)*(Ny+1)-1;
16
17  % beta の設定
18  nnode_y1=Ny+1;
```

```
19  beta=4*ones(dim_psi+1,1);
20  for i=2:2:2*Nx-2
21      beta(i*nnode_y1/2+1)=2;
22      beta(i*nnode_y1/2+Ny+1)=2;
23  end
24  for j=2:2:2*Ny-2
25      beta(j/2+1)=2;
26      beta(Nx*nnode_y1+j/2+1)=2;
27  end
28  beta(1)=1;
29  beta(Ny+1)=1;
30  beta(Nx*nnode_y1+1)=1;
31  beta(Nx*nnode_y1+Ny+1)=1;
32
33  % 行列 Lhat0 の作成
34  Lhat0=hx*hy/900*[16 -4  1 -4   8  -2  -2   8   4;
35                   -4 16 -4  1   8   8  -2  -2   4;
36                    1 -4 16 -4  -2   8   8  -2   4;
37                   -4  1 -4 16  -2  -2   8   8   4;
38                    8  8 -2 -2  64   4 -16   4  32;
39                   -2  8  8 -2   4  64   4 -16  32;
40                   -2 -2  8  8 -16   4  64   4  32;
41                    8 -2 -2  8   4 -16   4  64  32;
42                    4  4  4  4  32  32  32  32 256];
43  % 行列 Lhat の作成
44  Lhat=assem_interval1(Lhat0,Nx,Ny);
45
46  % 行列 L の作成
47  L=delete_boundary(Lhat,Nx,Ny);
48
49  % 行列 D00 の作成
50  D00=hx/(90*hy)*[28  -7  -1   4   14   8    2 -32  -16;
51                  -7  28   4  -1   14 -32    2   8  -16;
52                  -1   4  28  -7    2 -32   14   8  -16;
53                   4  -1  -7  28    2   8   14 -32  -16;
54                  14  14   2   2  112 -16   16 -16 -128;
55                   8 -32 -32   8  -16  64  -16 -16   32;
56                   2   2  14  14   16 -16  112 -16 -128;
57                 -32   8   8 -32  -16 -16  -16  64   32;
58                 -16 -16 -16 -16 -128  32 -128  32  256];
59  D00=D00+hy/(90*hx)*[28   4  -1  -7 -32    2   8   14  -16;
60                       4  28  -7  -1 -32   14   8    2  -16;
61                      -1  -7  28   4   8   14 -32    2  -16;
62                      -7  -1   4  28   8    2 -32   14  -16;
63                     -32 -32   8   8  64  -16 -16  -16   32;
64                       2  14  14   2 -16  112 -16   16 -128;
65                       8   8 -32 -32 -16  -16  64  -16   32;
66                      14   2   2  14 -16   16 -16  112 -128;
67                     -16 -16 -16 -16  32 -128  32 -128  256];
68
69  % 行列 D0 の作成
70  D0=delete_boundary(assem_interval1(D00,Nx,Ny),Nx,Ny);
71
72  % 行列 Dhatxx0 の作成
73  Dhatxx0=hy/(90*hx)*[28   4  -1  -7 -32    2   8   14  -16;
74                       4  28  -7  -1 -32   14   8    2  -16;
75                      -1  -7  28   4   8   14 -32    2  -16;
76                      -7  -1   4  28   8    2 -32   14  -16;
```

```
                    -32 -32   8   8  64  -16 -16  -16   32;
                      2  14  14   2 -16  112 -16   16 -128;
                      8   8 -32 -32 -16  -16  64  -16   32;
                     14   2   2  14 -16   16 -16  112 -128;
                    -16 -16 -16 -16  32 -128  32 -128  256];

% 行列 Dhatxx の作成
Dhatxx=assem_interval1(Dhatxx0,Nx,Ny);

% 行列 Dhatxy0 の作成
Dhatxy0=intval(1)/36*[9  -3  -1   3  12   4   4 -12 -16;
                      3  -9  -3   1 -12  12  -4  -4  16;
                     -1   3   9  -3   4 -12  12   4 -16;
                     -3   1   3  -9  -4  -4 -12  12  16;
                    -12  12   4  -4   0 -16   0  16   0;
                      4 -12  12  -4 -16   0  16   0   0;
                      4  -4 -12  12   0  16   0 -16   0;
                     12  -4   4 -12  16   0 -16   0   0;
                    -16  16 -16  16   0   0   0   0   0];

% 行列 Dhatxy の作成
Dhatxy=assem_interval1(Dhatxy0,Nx,Ny);

% 行列 Dhatyy0 の作成
Dhatyy0=hx/(90*hy)*[28  -7  -1   4   14   8    2 -32  -16;
                    -7  28   4  -1   14 -32    2   8  -16;
                    -1   4  28  -7    2 -32   14   8  -16;
                     4  -1  -7  28    2   8   14 -32  -16;
                    14  14   2   2  112 -16   16 -16 -128;
                     8 -32 -32   8  -16  64  -16 -16   32;
                     2   2  14  14   16 -16  112 -16 -128;
                   -32   8   8 -32  -16 -16  -16  64   32;
                   -16 -16 -16 -16 -128  32 -128  32 256];

% 行列 Dhatyy の作成
Dhatyy=assem_interval1(Dhatyy0,Nx,Ny);

% 行列 Dxx の作成
Dxx=delete_boundary(Dhatxx,Nx,Ny);

% 行列 Dxy の作成
Dxy=delete_boundary(Dhatxy,Nx,Ny);

% 行列 Dyy の作成
Dyy=delete_boundary(Dhatyy,Nx,Ny);

% 行列 Kx0 の作成
Kx0=hy/180*[-12   4  -1   3 -16   2   4  -6  -8;
             -4  12  -3   1  16   6  -4  -2   8;
              1  -3  12  -4  -4   6  16  -2   8;
              3  -1   4 -12   4   2 -16  -6  -8;
             16 -16   4  -4   0  -8   0   8   0;
             -2   6   6  -2   8  48   8 -16  64;
             -4   4 -16  16   0  -8   0   8   0;
             -6   2   2  -6  -8  16  -8 -48 -64;
              8  -8  -8   8   0 -64   0  64   0];

% 行列 Kx の作成
```

```
135 Kx=assem_interval2(Kx0,Nx,Ny);
136
137 % 行列 Ky0 の作成
138 Ky0=hx/180*[-12   3  -1  4   -6   4   2 -16  -8;
139               3 -12   4  -1  -6 -16   2   4  -8;
140               1  -4  12  -3  -2  16   6  -4   8;
141              -4   1  -3  12  -2  -4   6  16   8;
142              -6  -6   2   2 -48  -8  16  -8 -64;
143              -4  16 -16   4   8   0  -8   0   0;
144              -2  -2   6   6 -16   8  48   8  64;
145              16  -4   4 -16   8   0  -8   0   0;
146               8   8  -8  -8  64   0 -64   0   0];
147
148 % 行列 Ky の作成
149 Ky=assem_interval2(Ky0,Nx,Ny);
150
151 % 行列 Fhatxx0 の作成
152 Fhatxx0=hy/(6*hx)*[ 1 -1  0  0;
153                    -1  1  0  0;
154                     0  0  1 -1;
155                     0  0 -1  1;
156                     0  0  0  0;
157                    -2  2  2 -2;
158                     0  0  0  0;
159                     2 -2 -2  2;
160                     0  0  0  0];
161
162 % 行列 Fhatxx の作成
163 Fhatxx=assem_interval3(Fhatxx0,beta,Nx,Ny);
164
165 % 行列 Fhatxy0 の作成
166 Fhatxy0=intval(1)/36*[  5   1  -1  -5;
167                        -1  -5   5   1;
168                        -1  -5   5   1;
169                         5   1  -1  -5;
170                        -4   4  -4   4;
171                        -4 -20  20   4;
172                        -4   4  -4   4;
173                        20   4  -4 -20;
174                       -16  16 -16  16];
175
176 % 行列 Fhatxy の作成
177 Fhatxy=assem_interval3(Fhatxy0,beta,Nx,Ny);
178
179 % 行列 Fhatyx0 の作成
180 Fhatyx0=intval(1)/36*[  5  -5  -1   1;
181                         5  -5  -1   1;
182                        -1   1   5  -5;
183                        -1   1   5  -5;
184                        20 -20  -4   4;
185                        -4   4  -4   4;
186                        -4   4  20 -20;
187                        -4   4  -4   4;
188                       -16  16 -16  16];
189
190 % 行列 Fhatyx の作成
191 Fhatyx=assem_interval3(Fhatyx0,beta,Nx,Ny);
192
```

```
% 行列 Fhatyy0 の作成
Fhatyy0=hx/(6*hy)*[ 1  0  0 -1;
                    0  1 -1  0;
                    0 -1  1  0;
                   -1  0  0  1;
                    2  2 -2 -2;
                    0  0  0  0;
                   -2 -2  2  2;
                    0  0  0  0;
                    0  0  0  0];

% 行列 Fhatyy の作成
Fhatyy=assem_interval3(Fhatyy0,beta,Nx,Ny);

% 行列 Dtilde0 の作成
Dtilde0=1/(6*hx*hy)*[2*hx2+2*hy2    hx2-2*hy2     -hx2-hy2   -2*hx2+hy2;
                       hx2-2*hy2  2*hx2+2*hy2   -2*hx2+hy2     -hx2-hy2;
                        -hx2-hy2   -2*hx2+hy2  2*hx2+2*hy2    hx2-2*hy2;
                      -2*hx2+hy2     -hx2-hy2    hx2-2*hy2  2*hx2+2*hy2];

% 行列 Dtilde の作成
Dtilde=assem_interval4(Dtilde0,beta,Nx,Ny);

% 行列 Ex0 の作成
Ex0=hy/36*[-5 1 0  0  4  2  0 -10  8;
           -1 5 0  0 -4 10  0  -2 -8;
            0 0 5 -1  0 10 -4  -2 -8;
            0 0 1 -5  0  2  4 -10  8];

% 行列 Ex の作成
Ex=assem_interval5(Ex0,beta,Nx,Ny);

% 行列 Ey0 の作成
Ey0=hx/36*[-5  0 0 1 -10  0  2  4  8;
            0 -5 1 0 -10  4  2  0  8;
            0 -1 5 0  -2 -4 10  0 -8;
           -1  0 0 5  -2  0 10 -4 -8];

% 行列 Ey の作成
Ey=assem_interval5(Ey0,beta,Nx,Ny);

%                                  %
% 基底関数からの行列作成 ここまで %
%                                  %

% 行列 D の作成
D=[         nu*D0  zeros(dim_phi);
   zeros(dim_phi)          nu*D0];

% 行列 E の作成
E=[Ex Ey];

% 行列 G の作成
G=[ D            -E';
   -E zeros(dim_psi)];

% 行列 invG の作成
invG=inv(G);
```

```
251
252 % 行列 Ga, Gb, Gstar の作成
253 Ga=invG(1:2*dim_phi,1:2*dim_phi);
254 Gb=invG(2*dim_phi+1:2*dim_phi+dim_psi,1:2*dim_phi);
255 Gstar=invG(2*dim_phi+1:2*dim_phi+dim_psi,2*dim_phi+1:2*dim_phi+dim_psi);
256
257 % 行列 invLhat の作成
258 invLhat=inv(Lhat);
259
260 % 行列 Mx, My の作成
261 Mx=Kx*invLhat;
262 My=Ky*invLhat;
263
264 % 行列 invL の作成
265 invL=inv(L);
266
267 % 行列 F の作成
268 F=[         invL  zeros(dim_phi);
269    zeros(dim_phi)          invL];
270
271 % 行列 Q1 の作成
272 tmpQ1=D0-Mx*Kx'-My*Ky';
273 Q1=[         tmpQ1 zeros(dim_phi);
274     zeros(dim_phi)         tmpQ1];
275
276 % 行列 A1 の作成
277 A1=Ga*Q1*Ga;
278
279 % 行列 Ehatxx, Ehatxy, Ehatyy の作成
280 Ehatxx=Mx*Dhatxx*Mx';
281 Ehatxy=Mx*Dhatxy*My';
282 Ehatyy=My*Dhatyy*My';
283
284 % 行列 E1 の作成
285 tmpE1=Ehatxx+Ehatxy+Ehatxy'+Ehatyy;
286 E1=[         tmpE1 zeros(dim_phi);
287     zeros(dim_phi)         tmpE1];
288
289 % 行列 E2 の作成
290 E2=[(Mx*Fhatxx+My*Fhatyx)*Gb;
291     (Mx*Fhatxy+My*Fhatyy)*Gb];
292
293 % 行列 E3 の作成
294 tmpE3=-(Kx*invLhat*Kx'+Ky*invLhat*Ky')*invL;
295 E3=[         tmpE3  zeros(dim_phi);
296     zeros(dim_phi)         tmpE3];
297
298 % 行列 A2 の作成
299 A2=nu^2*Ga'*E1*Ga-nu*Ga*E2-nu*(Ga*E2)'+nu*Ga*E3+nu*(Ga*E3)'+Gb'*E*F+(Gb'*E*F)'+Gb'*Dtilde*Gb+F;
300
301 % 行列 Q3 の作成
302 Q3=[Dxx  Dxy;
303     Dxy' Dyy];
304
305 % 行列 A3 の作成
306 A3=Ga*Q3*Ga;
307
308 % 行列 A4 の作成
```

```
309  A4=Gb'*E*F+(Gb'*E*F)'+Gb'*Dtilde*Gb+F;
```

以下は apriori_interval.m から呼ばれる関数のプログラムである。

assem_interval1.m

```
 1  % assem_interval1.m
 2  % 要素係数行列から全体行列を作り出す (interval)
 3  % Lhat, Dxxhat, Dxyhat, Dyyhat に対応
 4
 5  function ret = assem_interval1(A0,Nx,Ny)
 6
 7  nnode_x=2*Nx+1;
 8  nnode_y=2*Ny+1;
 9
10  A=intval(zeros(nnode_x*nnode_y));
11
12  num=[0 2*nnode_y 2*nnode_y+2 2 nnode_y 2*nnode_y+1 nnode_y+2 1 nnode_y+1];
13
14  for p=1:Nx
15      for q=1:Ny
16          i=2*p-2;
17          j=2*q-2;
18          % l : 要素 e_pq の局所節点番号 1 の全体節点番号
19          l=i*nnode_y+j+1;
20          for s=1:9
21              for t=1:9
22                  A(l+num(s),l+num(t))= A(l+num(s),l+num(t))+A0(s,t);
23              end
24          end
25      end
26  end
27  ret=A;
```

assem_interval2.m

```
 1  % assem_interval2.m
 2  % 要素係数行列から全体行列を作り出す (interval)
 3  % Kx, Ky に対応
 4
 5  function ret = assem_interval2(A0,Nx,Ny)
 6
 7  nnode_x=2*Nx+1;
 8  nnode_y=2*Ny+1;
 9
10  size_A=nnode_x*nnode_y;
11
12  A=intval(zeros(size_A));
13
14  num=[0 2*nnode_y 2*nnode_y+2 2 nnode_y 2*nnode_y+1 nnode_y+2 1 nnode_y+1];
15
16  for p=1:Nx
17      for q=1:Ny
18          i=2*p-2;
19          j=2*q-2;
20          % l : 要素 e_pq の局所節点番号 1 の全体節点番号
21          l=i*nnode_y+j+1;
22          for s=1:9
```

```
23              for t=1:9
24                  A(l+num(s),l+num(t))= A(l+num(s),l+num(t))+A0(s,t);
25              end
26          end
27      end
28  end
29
30  count=1;
31  for n=nnode_y+2:size_A-nnode_y-1
32      if mod(n,nnode_y)~=0 && mod(n,nnode_y)~=1
33              nb_num(count)=n;
34          count=count+1;
35      end
36  end
37  A=A(nb_num,:);
38  ret=A;
```

assem_interval3.m

```
 1  % assem_interval3.m
 2  % 要素係数行列から全体行列を作り出す (interval)
 3  % Fhatxx, Fhatxy, Fhatyx, Fhatyy に対応
 4
 5  function ret = assem_interval3(A0,beta,Nx,Ny)
 6
 7  nnode_x=2*Nx+1;
 8  nnode_y=2*Ny+1;
 9
10  nnode_x1=Nx+1;
11  nnode_y1=Ny+1;
12
13  A=intval(zeros(nnode_x*nnode_y,nnode_x1*nnode_y1));
14
15  num=[0 2*nnode_y 2*nnode_y+2 2 nnode_y 2*nnode_y+1 nnode_y+2 1 nnode_y+1];
16  num1=[0 nnode_y1 nnode_y1+1 1];
17
18  for p=1:Nx
19      for q=1:Ny
20          i=2*p-2;
21          j=2*q-2;
22          % l : 要素 e_pq の局所節点番号 1 の Φ hat での全体節点番号
23          % m : 要素 e_pq の局所節点番号 1 の φ     での全体節点番号
24          l=i*nnode_y+j+1;
25          m=i*nnode_y1/2+j/2+1;
26          for s=1:9
27              for t=1:4
28                  A(l+num(s),m+num1(t))= A(l+num(s),m+num1(t))+A0(s,t);
29              end
30          end
31      end
32  end
33  for i=1:nnode_x*nnode_y
34      for j=1:nnode_x1*nnode_y1
35          A(i,j)=A(i,j)-beta(j)*A(i,nnode_x1*nnode_y1);
36      end
37  end
38  A(:,nnode_x1*nnode_y1)=[];
39  ret=A;
```

assem_interval4.m

```
 1  % assem_interval4.m
 2  % 要素係数行列から全体行列を作り出す (interval)
 3  % Dtilde に対応
 4
 5  function ret = assem_interval4(A0,beta,Nx,Ny)
 6
 7  nnode_x1=Nx+1;
 8  nnode_y1=Ny+1;
 9
10  A=intval(zeros(nnode_x1*nnode_y1));
11
12  num1=[0 nnode_y1 nnode_y1+1 1];
13
14  for p=1:Nx
15      for q=1:Ny
16          i=2*p-2;
17          j=2*q-2;
18          % m : 要素 e_pq の局所節点番号 1 の φ での全体節点番号
19          m=i*nnode_y1/2+j/2+1;
20          for s=1:4
21              for t=1:4
22                  A(m+num1(s),m+num1(t))=A(m+num1(s),m+num1(t))+A0(s,t);
23              end
24          end
25      end
26  end
27
28  for i=1:nnode_x1*nnode_y1
29      for j=1:nnode_x1*nnode_y1
30          m=nnode_x1*nnode_y1;
31          A(i,j)=A(i,j)-beta(i)*A(m,j)-beta(j)*A(m,i)+beta(i)*beta(j)*A(m,m);
32      end
33  end
34
35  A(nnode_x1*nnode_y1,:)=[];
36  A(:,nnode_x1*nnode_y1)=[];
37  ret=A;
```

assem_interval5.m

```
 1  % assem_interval5.m
 2  % 要素係数行列から全体行列を作り出す (interval)
 3  % Ex, Ey に対応
 4
 5  function ret = assem_interval5(A0,beta,Nx,Ny)
 6
 7  nnode_x=2*Nx+1;
 8  nnode_y=2*Ny+1;
 9
10  size_A=nnode_x*nnode_y;
11
12  nnode_x1=Nx+1;
13  nnode_y1=Ny+1;
14
15  A=intval(zeros(nnode_x1*nnode_y1,nnode_x*nnode_y));
```

```
16
17  num=[0 2*nnode_y 2*nnode_y+2 2 nnode_y 2*nnode_y+1 nnode_y+2 1 nnode_y+1];
18  num1=[0 nnode_y1 nnode_y1+1 1];
19
20  for p=1:Nx
21      for q=1:Ny
22          i=2*p-2;
23          j=2*q-2;
24          % l : 要素 e_pq の局所節点番号 1 の Φ hat での全体節点番号
25          % m : 要素 e_pq の局所節点番号 1 の φ     での全体節点番号
26          l=i*nnode_y+j+1;
27          m=i*nnode_y1/2+j/2+1;
28          for s=1:4
29              for t=1:9
30                  A(m+num1(s),l+num(t))= A(m+num1(s),l+num(t))+A0(s,t);
31              end
32          end
33      end
34  end
35
36  for i=1:nnode_x1*nnode_y1
37      for j=1:nnode_x*nnode_y
38          A(i,j)=A(i,j)-beta(i)*A(nnode_x1*nnode_y1,j);
39      end
40  end
41
42  count=1;
43  for n=nnode_y+2:size_A-nnode_y-1
44      if mod(n,nnode_y)~=0 && mod(n,nnode_y)~=1
45              nb_num(count)=n;
46          count=count+1;
47      end
48  end
49  A=A(:,nb_num);
50  A(nnode_x1*nnode_y1,:)=[];
51  ret=A;
```

9.2.4 アプリオリ誤差定数を求める実験用のプログラム

apriori_test_floating.m

```
 1  % apriori_test_ floating.m
 2  % アプリオリ誤差定数を floating で求める
 3  % 実験用のプログラム
 4
 5  clear
 6
 7  format long e
 8
 9  Nx=5;
10  Ny=5;
11
12  nu=1;
13
14  width=1;
15  height=1;
```

```
16
17  C0=1/(2*pi);
18
19  h=max(1/Nx,1/Ny);
20
21  fprintf('行列作成開始\n');
22  tic
23  [F,A1,A2,A3,A4]=apriori_floating(Nx,Ny,nu,width,height);
24  matrix_time=toc;
25  fprintf('行列作成終了\n 行列作成に %f 秒かかりました\n',matrix_time);
26
27  F=1/2*(F+F');
28
29  opts.disp=0;
30  tic
31  K1=sqrt(abs(eigs(A1,F,1,'lm',opts)))
32  K1_time=toc;
33  fprintf('K1 を求めるのに %f 秒かかりました\n\n',K1_time);
34
35  tic
36  K2=sqrt(abs(eigs(A2,F,1,'lm',opts)))
37  K2_time=toc;
38  fprintf('K2 を求めるのに %f 秒かかりました\n\n',K2_time);
39
40  tic
41  K3=sqrt(abs(eigs(A3,F,1,'lm',opts)))
42  K3_time=toc;
43  fprintf('K3 を求めるのに %f 秒かかりました\n\n',K3_time);
44
45  tic
46  K4=sqrt(abs(eigs(A4,F,1,'lm',opts)))
47  K4_time=toc;
48  fprintf('K4 を求めるのに %f 秒かかりました\n\n',K4_time);
49
50  Ch1=sqrt((nu*K1+C0*h*K2+K3)^2+(C0*h)^2)
51  Ch2=sqrt((C0*h*K4+K3)^2+(C0*h)^2)
52
53  beta=1/sqrt(4+2*sqrt(2));
54  v_constant1=sqrt(1/(nu^2)+1/(beta^2))*Ch1
55  p_constant1=(1/beta+nu/(beta^2))*Ch1
56
57  v_constant2=sqrt(1/(nu^2)+1/(beta^2))*Ch2
58  p_constant2=(1/beta+nu/(beta^2))*Ch2
59
60  L2_error=nu*v_constant2^2+2*p_constant2*K3
```

apriori_test_interval.m

```
1  % apriori_test_interval.m
2  % アプリオリ誤差定数を interval で求める
3  % 実験用のプログラム
4
5  clear
6
7  format long e
8
9  % G_RUMP : 1 , ADM_a : 2
```

```
10  frag=2;
11
12
13  Nx=5;
14  Ny=5;
15
16  nu=1;
17
18  delta=1e-9;
19  ddelta=1e-9;
20
21  VPDdelta=1e-2;
22  VPDddelta=1e-2;
23
24  width=1;
25  height=1;
26
27  C0=intval(1)/(2*pi);
28  beta=intval(1)/sqrt(4+2*sqrt(2));
29
30  h=max(sup(intval(width)/Nx),sup(intval(height)/Ny));
31
32  fprintf('行列作成開始\n');
33  tic
34  [intF,intA1,intA2,intA3,intA4]=apriori_interval(Nx,Ny,nu,width,height);
35  matrix_time=toc;
36  fprintf('行列作成終了\n行列作成に %f 秒かかりました\n',matrix_time);
37
38  if frag==1
39  tic
40  G_RUMP_K1=sqrt(G_RUMP(delta,ddelta,VPDdelta,VPDddelta,intA1,intF));
41  G_RUMP_K1_time=toc;
42  fprintf('\n K1 を求めるのに %f 秒かかりました\n',G_RUMP_K1_time);
43  infsup(G_RUMP_K1)
44
45  tic
46  G_RUMP_K2=sqrt(G_RUMP(delta,ddelta,VPDdelta,VPDddelta,intA2,intF));
47  G_RUMP_K2_time=toc;
48  fprintf('\n K2 を求めるのに %f 秒かかりました\n',G_RUMP_K2_time);
49  infsup(G_RUMP_K2)
50
51  tic
52  G_RUMP_K3=sqrt(G_RUMP(delta,ddelta,VPDdelta,VPDddelta,intA3,intF));
53  G_RUMP_K3_time=toc;
54  fprintf('\n K3 を求めるのに %f 秒かかりました\n',G_RUMP_K3_time);
55  infsup(G_RUMP_K3)
56
57  tic
58  G_RUMP_K4=sqrt(G_RUMP(delta,ddelta,VPDdelta,VPDddelta,intA4,intF));
59  G_RUMP_K4_time=toc;
60  fprintf('\n K4 を求めるのに %f 秒かかりました\n',G_RUMP_K4_time);
61  infsup(G_RUMP_K4)
62  fprintf('\n');
63
64  G_RUMP_Ch1=sqrt((nu*G_RUMP_K1+C0*h*G_RUMP_K2+G_RUMP_K3)^2+(C0*h)^2);
65  infsup(G_RUMP_Ch1)
66  G_RUMP_Ch2=sqrt((C0*h*G_RUMP_K4+G_RUMP_K3)^2+(C0*h)^2);
67  infsup(G_RUMP_Ch2)
```

```
 68
 69  nu=intval(nu);
 70  v_constant1=sqrt(1/(nu^2)+1/(beta^2))*G_RUMP_Ch1;
 71  p_constant1=(1/beta+nu/(beta^2))*G_RUMP_Ch1;
 72  v_constant2=sqrt(1/(nu^2)+1/(beta^2))*G_RUMP_Ch2;
 73  p_constant2=(1/beta+nu/(beta^2))*G_RUMP_Ch2;
 74  infsup(v_constant1)
 75  infsup(p_constant1)
 76  infsup(v_constant2)
 77  infsup(p_constant2)
 78
 79  G_RUMP_L2_error=nu*v_constant2^2+2*p_constant2*G_RUMP_K3;
 80  infsup(G_RUMP_L2_error)
 81  end
 82
 83  if frag==2
 84  tic
 85  ADM_a_K1=sqrt(ADM_a(intA1,intF));
 86  ADM_a_K1_time=toc;
 87  fprintf('\n K1 を求めるのに %f 秒かかりました\n',ADM_a_K1_time);
 88  infsup(ADM_a_K1)
 89
 90  tic
 91  ADM_a_K2=sqrt(ADM_a(intA2,intF));
 92  ADM_a_K2_time=toc;
 93  fprintf('\n K2 を求めるのに %f 秒かかりました\n',ADM_a_K2_time);
 94  infsup(ADM_a_K2)
 95
 96  tic
 97  ADM_a_K3=sqrt(ADM_a(intA3,intF));
 98  ADM_a_K3_time=toc;
 99  fprintf('\n K3 を求めるのに %f 秒かかりました\n',ADM_a_K3_time);
100  infsup(ADM_a_K3)
101
102  tic
103  ADM_a_K4=sqrt(ADM_a(intA4,intF));
104  ADM_a_K4_time=toc;
105  fprintf('\n K4 を求めるのに %f 秒かかりました\n',ADM_a_K4_time);
106  infsup(ADM_a_K4)
107  fprintf('\n');
108
109  ADM_a_Ch1=sqrt((nu*ADM_a_K1+C0*h*ADM_a_K2+ADM_a_K3)^2+(C0*h)^2);
110  infsup(ADM_a_Ch1)
111  ADM_a_Ch2=sqrt((C0*h*ADM_a_K4+ADM_a_K3)^2+(C0*h)^2);
112  infsup(ADM_a_Ch2)
113
114  nu=intval(nu);
115  v_constant1=sqrt(1/(nu^2)+1/(beta^2))*ADM_a_Ch1;
116  p_constant1=(1/beta+nu/(beta^2))*ADM_a_Ch1;
117  v_constant2=sqrt(1/(nu^2)+1/(beta^2))*ADM_a_Ch2;
118  p_constant2=(1/beta+nu/(beta^2))*ADM_a_Ch2;
119  infsup(v_constant1)
120  infsup(p_constant1)
121  infsup(v_constant2)
122  infsup(p_constant2)
123
124  ADM_a_L2_error=nu*v_constant2^2+2*p_constant2*ADM_a_K3;
125  infsup(ADM_a_L2_error)
```

```
126  end
```

9.3　事後誤差評価の数値実験で用いたプログラム

9.3.1　floating で事後誤差定数を求めるプログラム

以下は floating で事後誤差定数と流速と圧力に関する事後誤差限界を求めるプログラムである。

aposteriori_floating.m

```
 1  % aposteriori_floating.m
 2  % 事後誤差定数 C(uh,ph) と |div uh|_0 を floating で求め
 3  % 外力密度 f と有限要素解 uh,ph のグラフを描く
 4
 5  function [C_uh_ph,div_uh_0]=aposteriori_floating(Nx,Ny,nu,width,height);
 6
 7  hx=width/Nx;
 8  hy=height/Ny;
 9
10  hx2=hx^2;
11  hy2=hy^2;
12
13  dim_phihat=(2*Nx+1)*(2*Ny+1);
14  dim_phi=(2*Nx-1)*(2*Ny-1);
15  dim_psi=(Nx+1)*(Ny+1)-1;
16
17  % beta の設定
18  nnode_y1=Ny+1;
19  beta=4*ones(dim_psi+1,1);
20  for i=2:2:2*Nx-2
21      beta(i*nnode_y1/2+1)=2;
22      beta(i*nnode_y1/2+Ny+1)=2;
23  end
24  for j=2:2:2*Ny-2
25      beta(j/2+1)=2;
26      beta(Nx*nnode_y1+j/2+1)=2;
27  end
28  beta(1)=1;
29  beta(Ny+1)=1;
30  beta(Nx*nnode_y1+1)=1;
31  beta(Nx*nnode_y1+Ny+1)=1;
32
33  % 行列 Lhat0 の作成
34  Lhat0=hx*hy/900*[16 -4  1 -4   8  -2  -2   8   4;
35                   -4 16 -4  1   8   8  -2  -2   4;
36                    1 -4 16 -4  -2   8   8  -2   4;
37                   -4  1 -4 16  -2  -2   8   8   4;
38                    8  8 -2 -2  64   4 -16   4  32;
39                   -2  8  8 -2   4  64   4 -16  32;
40                   -2 -2  8  8 -16   4  64   4  32;
41                    8 -2 -2  8   4 -16   4  64  32;
42                    4  4  4  4  32  32  32  32 256];
43  % 行列 Lhat の作成
44  Lhat=assem_floating1(Lhat0,Nx,Ny);
```

```
45
46 % 行列 L の作成
47 L=delete_boundary(Lhat,Nx,Ny);
48
49 % 行列 D00 の作成
50 D00=hx/(90*hy)*[28  -7  -1   4   14   8    2 -32  -16;
51                 -7  28   4  -1   14 -32    2   8  -16;
52                 -1   4  28  -7    2 -32   14   8  -16;
53                  4  -1  -7  28    2   8   14 -32  -16;
54                 14  14   2   2  112 -16   16 -16 -128;
55                  8 -32 -32   8  -16  64  -16 -16   32;
56                  2   2  14  14   16 -16  112 -16 -128;
57                -32   8   8 -32  -16 -16  -16  64   32;
58                -16 -16 -16 -16 -128  32 -128  32  256];
59 D00=D00+hy/(90*hx)*[28   4  -1  -7 -32    2   8   14  -16;
60                      4  28  -7  -1 -32   14   8    2  -16;
61                     -1  -7  28   4   8   14 -32    2  -16;
62                     -7  -1   4  28   8    2 -32   14  -16;
63                    -32 -32   8   8  64  -16 -16  -16   32;
64                      2  14  14   2 -16  112 -16   16 -128;
65                      8   8 -32 -32 -16  -16  64  -16   32;
66                     14   2   2  14 -16   16 -16  112 -128;
67                    -16 -16 -16 -16  32 -128  32 -128  256];
68
69 % 行列 D0 の作成
70 D0=delete_boundary(assem_floating1(D00,Nx,Ny),Nx,Ny);
71
72 % 行列 Dhatxx0 の作成
73 Dhatxx0=hy/(90*hx)*[28   4  -1  -7 -32    2   8   14  -16;
74                      4  28  -7  -1 -32   14   8    2  -16;
75                     -1  -7  28   4   8   14 -32    2  -16;
76                     -7  -1   4  28   8    2 -32   14  -16;
77                    -32 -32   8   8  64  -16 -16  -16   32;
78                      2  14  14   2 -16  112 -16   16 -128;
79                      8   8 -32 -32 -16  -16  64  -16   32;
80                     14   2   2  14 -16   16 -16  112 -128;
81                    -16 -16 -16 -16  32 -128  32 -128  256];
82
83 % 行列 Dhatxx の作成
84 Dhatxx=assem_floating1(Dhatxx0,Nx,Ny);
85
86 % 行列 Dhatxy0 の作成
87 Dhatxy0=1/36*[9  -3  -1   3  12   4   4 -12 -16;
88                3  -9  -3   1 -12  12  -4  -4  16;
89               -1   3   9  -3   4 -12  12   4 -16;
90               -3   1   3  -9  -4  -4 -12  12  16;
91              -12  12   4  -4   0 -16   0  16   0;
92                4 -12  12  -4 -16   0  16   0   0;
93                4  -4 -12  12   0  16   0 -16   0;
94               12  -4   4 -12  16   0 -16   0   0;
95              -16  16 -16  16   0   0   0   0   0];
96
97 % 行列 Dhatxy の作成
98 Dhatxy=assem_floating1(Dhatxy0,Nx,Ny);
99
100 % 行列 Dhatyy0 の作成
101 Dhatyy0=hx/(90*hy)*[28  -7  -1   4   14   8    2 -32  -16;
102                      -7  28   4  -1   14 -32    2   8  -16;
```

```
                    -1   4  28  -7    2 -32   14   8  -16;
                     4  -1  -7  28    2   8   14 -32  -16;
                    14  14   2   2  112 -16   16 -16 -128;
                     8 -32 -32   8  -16  64  -16 -16   32;
                     2   2  14  14   16 -16  112 -16 -128;
                   -32   8   8 -32  -16 -16  -16  64   32;
                   -16 -16 -16 -16 -128  32 -128  32 256];

% 行列 Dhatyy の作成
Dhatyy=assem_floating1(Dhatyy0,Nx,Ny);

% 行列 Dxx の作成
Dxx=delete_boundary(Dhatxx,Nx,Ny);

% 行列 Dxy の作成
Dxy=delete_boundary(Dhatxy,Nx,Ny);

% 行列 Dyy の作成
Dyy=delete_boundary(Dhatyy,Nx,Ny);

% 行列 Kx0 の作成
Kx0=hy/180*[-12   4  -1   3 -16   2   4  -6  -8;
             -4  12  -3   1  16   6  -4  -2   8;
              1  -3  12  -4  -4   6  16  -2   8;
              3  -1   4 -12   4   2 -16  -6  -8;
             16 -16   4  -4   0  -8   0   8   0;
             -2   6   6  -2   8  48   8 -16  64;
             -4   4 -16  16   0  -8   0   8   0;
             -6   2   2  -6  -8  16  -8 -48 -64;
              8  -8  -8   8   0 -64   0  64   0];

% 行列 Kx の作成
Kx=assem_floating2(Kx0,Nx,Ny);

% 行列 Ky0 の作成
Ky0=hx/180*[-12   3  -1  4   -6   4   2 -16  -8;
             3 -12   4  -1  -6 -16   2   4  -8;
             1  -4  12  -3  -2  16   6  -4   8;
            -4   1  -3  12  -2  -4   6  16   8;
            -6  -6   2   2 -48  -8  16  -8 -64;
            -4  16 -16   4   8   0  -8   0   0;
            -2  -2   6   6 -16   8  48   8  64;
            16  -4   4 -16   8   0  -8   0   0;
             8   8  -8  -8  64   0 -64   0   0];

% 行列 Ky の作成
Ky=assem_floating2(Ky0,Nx,Ny);

% 行列 Fhatxx0 の作成
Fhatxx0=hy/(6*hx)*[ 1 -1  0  0;
                   -1  1  0  0;
                    0  0  1 -1;
                    0  0 -1  1;
                    0  0  0  0;
                   -2  2  2 -2;
                    0  0  0  0;
                    2 -2 -2  2;
                    0  0  0  0];
```

```
161
162 % 行列 Fhatxx の作成
163 Fhatxx=assem_floating3(Fhatxx0,beta,Nx,Ny);
164
165 % 行列 Fhatxy0 の作成
166 Fhatxy0=1/36*[  5   1  -1  -5;
167               -1  -5   5   1;
168               -1  -5   5   1;
169                5   1  -1  -5;
170               -4   4  -4   4;
171               -4 -20  20   4;
172               -4   4  -4   4;
173               20   4  -4 -20;
174              -16  16 -16  16];
175
176 % 行列 Fhatxy の作成
177 Fhatxy=assem_floating3(Fhatxy0,beta,Nx,Ny);
178
179 % 行列 Fhatyx0 の作成
180 Fhatyx0=1/36*[  5  -5  -1   1;
181                 5  -5  -1   1;
182               -1   1   5  -5;
183               -1   1   5  -5;
184               20 -20  -4   4;
185               -4   4  -4   4;
186               -4   4  20 -20;
187               -4   4  -4   4;
188              -16  16 -16  16];
189
190 % 行列 Fhatyx の作成
191 Fhatyx=assem_floating3(Fhatyx0,beta,Nx,Ny);
192
193 % 行列 Fhatyy0 の作成
194 Fhatyy0=hx/(6*hy)*[ 1  0  0 -1;
195                     0  1 -1  0;
196                     0 -1  1  0;
197                    -1  0  0  1;
198                     2  2 -2 -2;
199                     0  0  0  0;
200                    -2 -2  2  2;
201                     0  0  0  0;
202                     0  0  0  0];
203
204 % 行列 Fhatyy の作成
205 Fhatyy=assem_floating3(Fhatyy0,beta,Nx,Ny);
206
207 % 行列 Dtilde0 の作成
208 Dtilde0=1/(6*hx*hy)*[2*hx2+2*hy2    hx2-2*hy2     -hx2-hy2   -2*hx2+hy2;
209                       hx2-2*hy2  2*hx2+2*hy2   -2*hx2+hy2     -hx2-hy2;
210                        -hx2-hy2   -2*hx2+hy2  2*hx2+2*hy2    hx2-2*hy2;
211                      -2*hx2+hy2     -hx2-hy2    hx2-2*hy2  2*hx2+2*hy2];
212
213 % 行列 Dtilde の作成
214 Dtilde=assem_floating4(Dtilde0,beta,Nx,Ny);
215
216 % 行列 Ex0 の作成
217 Ex0=hy/36*[-5 1 0  0  4  2  0 -10  8;
218            -1 5 0  0 -4 10  0  -2 -8;
```

```
219             0 0 5 -1  0 10 -4  -2 -8;
220             0 0 1 -5  0  2  4 -10  8];
221
222 % 行列 Ex の作成
223 Ex=assem_floating5(Ex0,beta,Nx,Ny);
224
225 % 行列 Ey0 の作成
226 Ey0=hx/36*[-5  0 0 1 -10  0  2  4  8;
227             0 -5 1 0 -10  4  2  0  8;
228             0 -1 5 0  -2 -4 10  0 -8;
229            -1  0 0 5  -2  0 10 -4 -8];
230
231 % 行列 Ey の作成
232 Ey=assem_floating5(Ey0,beta,Nx,Ny);
233
234 %                                   %
235 % 基底関数からの行列作成 ここまで %
236 %                                   %
237
238 % 行列 D の作成
239 D=[         nu*D0  zeros(dim_phi);
240    zeros(dim_phi)          nu*D0];
241
242 % 行列 E の作成
243 E=[Ex Ey];
244
245 % 行列 G の作成
246 G=[ D            -E';
247    -E zeros(dim_psi)];
248
249 % 行列 invG の作成
250 invG=inv(G);
251
252 % 行列 Ga, Gb, Gstar の作成
253 Ga=invG(1:2*dim_phi,1:2*dim_phi);
254 Gb=invG(2*dim_phi+1:2*dim_phi+dim_psi,1:2*dim_phi);
255 Gstar=invG(2*dim_phi+1:2*dim_phi+dim_psi,2*dim_phi+1:2*dim_phi+dim_psi);
256
257 % 行列 invLhat の作成
258 invLhat=inv(Lhat);
259
260 % 行列 Mx, My の作成
261 Mx=Kx*invLhat;
262 My=Ky*invLhat;
263
264 % 行列 invL の作成
265 invL=inv(L);
266
267 % 行列 F の作成
268 F=[         invL  zeros(dim_phi);
269    zeros(dim_phi)          invL];
270
271 % 行列 Q1 の作成
272 tmpQ1=D0-Mx*Kx'-My*Ky';
273 Q1=[         tmpQ1 zeros(dim_phi);
274     zeros(dim_phi)         tmpQ1];
275
276 % 行列 A1 の作成
```

```
277 A1=Ga*Q1*Ga;
278
279 % 行列 Ehatxx, Ehatxy, Ehatyy の作成
280 Ehatxx=Mx*Dhatxx*Mx';
281 Ehatxy=Mx*Dhatxy*My';
282 Ehatyy=My*Dhatyy*My';
283
284 % 行列 E1 の作成
285 tmpE1=Ehatxx+Ehatxy+Ehatxy'+Ehatyy;
286 E1=[         tmpE1 zeros(dim_phi);
287     zeros(dim_phi)         tmpE1];
288
289 % 行列 E2 の作成
290 E2=[(Mx*Fhatxx+My*Fhatyx)*Gb;
291     (Mx*Fhatxy+My*Fhatyy)*Gb];
292
293 % 行列 E3 の作成
294 tmpE3=-(Kx*invLhat*Kx'+Ky*invLhat*Ky')*invL;
295 tmpE3_2=-(Kx*invLhat*Kx'-Ky*invLhat*Ky')*invL;
296 E3=[        tmpE3  zeros(dim_phi);
297     zeros(dim_phi)         tmpE3];
298
299 % 行列 Q3 の作成
300 Q3=[Dxx  Dxy;
301     Dxy' Dyy];
302
303 % 行列 A3 の作成
304 A3=Ga*Q3*Ga;
305
306 % ベクトル f の作成
307 f1tilde=zeros(dim_phihat,1);
308 f2tilde=zeros(dim_phihat,1);
309
310 for i=0:2*Nx
311     for j=0:2*Ny
312         n=i*(2*Ny+1)+(j+1);
313         x=i*hx/2;
314         y=j*hy/2;
315         coordinates(n,1)=x;
316         coordinates(n,2)=y;
317         f1tilde(n)=f1_floating(x,y);
318         f2tilde(n)=f2_floating(x,y);
319     end
320 end
321
322 f1hat=Lhat*f1tilde;
323 f2hat=Lhat*f2tilde;
324
325 count=1;
326 for i=1+(2*Ny+1):dim_phihat-(2*Ny+1)
327     if mod(i,2*Ny+1)~=1 & mod(i,2*Ny+1)~=0
328        nb_num(count)=i;
329        count=count+1;
330     end
331 end
332
333 f1=f1hat(nb_num,:);
334 f2=f2hat(nb_num,:);
```

```
335
336 f=[f1;f2];
337
338 % C(uh,ph) の計算
339 C0=1/(2*pi);
340 h=max(hx,hy);
341
342 % 行列 Ehatx0 の作成
343 Ehatx0=hy/36*[-1 -1  0  0 -4 -2  0 -2 -8;
344                1  1  0  0  4  2  0  2  8;
345                0  0  1  1  0  2  4  2  8;
346                0  0 -1 -1  0 -2 -4 -2 -8];
347 % 行列 Ehaty0 の作成
348 Ehaty0=hx/36*[-1  0  0 -1 -2  0 -2 -4 -8;
349                0 -1 -1  0 -2 -4 -2  0 -8;
350                0  1  1  0  2  4  2  0  8;
351                1  0  0  1  2  0  2  4  8];
352
353 % 行列 Ehatx の作成
354 Ehatx=assem_floating6(Ehatx0,beta,Nx,Ny);
355
356 % 行列 Ehaty の作成
357 Ehaty=assem_floating6(Ehaty0,beta,Nx,Ny);
358
359 % 行列 Ehat の作成
360 Ehat=Ehatx*f1tilde+Ehaty*f2tilde;
361
362 % 行列 Kxhat の作成
363 Kxhat=assem_floating1(Kx0,Nx,Ny);
364
365 % 行列 Kyhat の作成
366 Kyhat=assem_floating1(Ky0,Nx,Ny);
367
368 % 行列 E4 の作成
369 tmpE4=Mx*Kxhat+My*Kyhat;
370 E4=[tmpE4                      zeros(dim_phi,dim_phihat);
371     zeros(dim_phi,dim_phihat)                    tmpE4];
372
373 % C(uh,pf) の作成
374 ftilde=[f1tilde;f2tilde];
375
376 lap_uh_lap_uh=f'*Ga'*E1*Ga*f;
377 lap_uh_nabla_ph=f'*Ga*E2*f;
378 lap_uh_f=f'*Ga*E4*ftilde;
379 nabla_ph_f=f'*Gb'*Ehat;
380 nabla_ph2=f'*Gb'*Dtilde*Gb*f;
381 norm_f2=f1tilde'*Lhat*f1tilde+f2tilde'*Lhat*f2tilde;
382
383 apo_A2=nu^2*lap_uh_lap_uh-2*nu*lap_uh_nabla_ph+2*nu*lap_uh_f-2*nabla_ph_f+nabla_ph2+norm_f2;
384
385 div_uh_0=sqrt(f'*A3*f);
386 C_uh_ph=nu*sqrt(f'*A1*f)+C0*h*sqrt(apo_A2)+div_uh_0;
387
388 % 以下グラフの作成
389 a=(Ga*f)';
390 b=(Gb*f)';
391
392 % 外力密度の表示
```

```
393 figure(1)
394 quiver(coordinates(:,1),coordinates(:,2),f1tilde,f2tilde)
395 axis equal
396 axis([0,width,0,height]);
397 title('A density of body forces')
398 xlabel('x')
399 ylabel('y')
400
401 % 流速のベクトル場を表示
402 coordinates2(:,1)=coordinates(nb_num,1);
403 coordinates2(:,2)=coordinates(nb_num,2);
404
405 figure(2)
406 subplot(1,2,1)
407 quiver(coordinates2(:,1),coordinates2(:,2),a(1,1:dim_phi),a(1,dim_phi+1:2*dim_phi),'b')
408 axis equal
409 axis([0,width,0,height])
410 title('Vector field(近似解)')
411 xlabel('x')
412 ylabel('y')
413
414 % 圧力場の表示
415 bm=0;
416 for i=1:dim_psi
417     bm=bm-b(i)*beta(i);
418 end
419 b=[b bm];
420
421 b1=zeros((Ny+1),(Nx+1));
422 b1(:)=b;
423 x_axis=0:width/Nx:width;
424 y_axis=0:height/Ny:height;
425 figure(2)
426 subplot(1,2,2)
427 meshc(x_axis,y_axis,b1)
428 axis square
429 title('Pressure field(近似解)')
430 xlabel('x')
431 ylabel('y')
432 zlabel('z')
```

以下は aposteriori_floating.m から呼ばれる関数のプログラムである。

assem_floating6.m

```
 1 % assem_floating6.m
 2 % 要素係数行列から全体行列を作り出す (floating)
 3 % Ehatx, Ehaty に対応
 4
 5 function ret = assem_floating6(A0,beta,Nx,Ny)
 6
 7 nnode_x=2*Nx+1;
 8 nnode_y=2*Ny+1;
 9
10 size_A=nnode_x*nnode_y;
11
12 nnode_x1=Nx+1;
13 nnode_y1=Ny+1;
```

```
14
15  A=zeros(nnode_x1*nnode_y1,nnode_x*nnode_y);
16
17  num=[0 2*nnode_y 2*nnode_y+2 2 nnode_y 2*nnode_y+1 nnode_y+2 1 nnode_y+1];
18  num1=[0 nnode_y1 nnode_y1+1 1];
19
20  for p=1:Nx
21      for q=1:Ny
22          i=2*p-2;
23          j=2*q-2;
24          % l : 要素 e_pq の局所節点番号 1 の Φ hat での全体節点番号
25          % m : 要素 e_pq の局所節点番号 1 の φ     での全体節点番号
26          l=i*nnode_y+j+1;
27          m=i*nnode_y1/2+j/2+1;
28          for s=1:4
29              for t=1:9
30                  A(m+num1(s),l+num(t))= A(m+num1(s),l+num(t))+A0(s,t);
31              end
32          end
33      end
34  end
35
36  for i=1:nnode_x1*nnode_y1
37      for j=1:nnode_x*nnode_y
38          A(i,j)=A(i,j)-beta(i)*A(nnode_x1*nnode_y1,j);
39      end
40  end
41
42  A(nnode_x1*nnode_y1,:)=[];
43  ret=A;
```

f1_floating.m

```
1  % 外力密度 f1_floating(x,y)
2  % floating 用
3
4  function ret = f1_floating(x,y)
5  ret=50*(-2*x+y+x*y);
```

f2_floating.m

```
1  % 外力密度 f2_floating(x,y)
2  % floating 用
3
4  function ret = f2_floating(x,y)
5  ret=20*(1-5*x*y);
```

9.3.2 interval で事後誤差定数を求めるプログラム

以下は interval で事後誤差定数と流速と圧力に関する事後誤差限界を求めるプログラムである。

aposteriori_interval.m

```
% aposteriori_interval.m
% 事後誤差定数 C(uh,ph) と |div uh|_0 を interval で求める

function [C_uh_ph,div_uh_0]=aposteriori_interval(Nx,Ny,nu,width,height);

hx=intval(width)/Nx;
hy=intval(height)/Ny;

hx2=hx^2;
hy2=hy^2;

dim_phihat=(2*Nx+1)*(2*Ny+1);
dim_phi=(2*Nx-1)*(2*Ny-1);
dim_psi=(Nx+1)*(Ny+1)-1;

% beta の設定
nnode_y1=Ny+1;
beta=4*ones(dim_psi+1,1);
for i=2:2:2*Nx-2
    beta(i*nnode_y1/2+1)=2;
    beta(i*nnode_y1/2+Ny+1)=2;
end
for j=2:2:2*Ny-2
    beta(j/2+1)=2;
    beta(Nx*nnode_y1+j/2+1)=2;
end
beta(1)=1;
beta(Ny+1)=1;
beta(Nx*nnode_y1+1)=1;
beta(Nx*nnode_y1+Ny+1)=1;

% 行列 Lhat0 の作成
Lhat0=hx*hy/900*[16 -4  1 -4   8  -2  -2   8   4;
                 -4 16 -4  1   8   8  -2  -2   4;
                  1 -4 16 -4  -2   8   8  -2   4;
                 -4  1 -4 16  -2  -2   8   8   4;
                  8  8 -2 -2  64   4 -16   4  32;
                 -2  8  8 -2   4  64   4 -16  32;
                 -2 -2  8  8 -16   4  64   4  32;
                  8 -2 -2  8   4 -16   4  64  32;
                  4  4  4  4  32  32  32  32 256];
% 行列 Lhat の作成
Lhat=assem_interval1(Lhat0,Nx,Ny);

% 行列 L の作成
L=delete_boundary(Lhat,Nx,Ny);

% 行列 D00 の作成
D00=hx/(90*hy)*[28  -7  -1   4   14   8    2 -32  -16;
                -7  28   4  -1   14 -32    2   8  -16;
                -1   4  28  -7    2 -32   14   8  -16;
                 4  -1  -7  28    2   8   14 -32  -16;
                14  14   2   2  112 -16   16 -16 -128;
                 8 -32 -32   8  -16  64  -16 -16   32;
                 2   2  14  14   16 -16  112 -16 -128;
               -32   8   8 -32  -16 -16  -16  64   32;
               -16 -16 -16 -16 -128  32 -128  32  256];
D00=D00+hy/(90*hx)*[28   4  -1  -7 -32    2   8   14  -16;
```

```
 59                       4  28  -7  -1 -32   14   8    2  -16;
 60                      -1  -7  28   4   8   14 -32    2  -16;
 61                      -7  -1   4  28   8    2 -32   14  -16;
 62                     -32 -32   8   8  64  -16 -16  -16   32;
 63                       2  14  14   2 -16  112 -16   16 -128;
 64                       8   8 -32 -32 -16  -16  64  -16   32;
 65                      14   2   2  14 -16   16 -16  112 -128;
 66                     -16 -16 -16 -16  32 -128  32 -128  256];
 67
 68  % 行列 D0 の作成
 69  D0=delete_boundary(assem_interval1(D00,Nx,Ny),Nx,Ny);
 70
 71  % 行列 Dhatxx0 の作成
 72  Dhatxx0=hy/(90*hx)*[28   4  -1  -7 -32    2   8   14  -16;
 73                       4  28  -7  -1 -32   14   8    2  -16;
 74                      -1  -7  28   4   8   14 -32    2  -16;
 75                      -7  -1   4  28   8    2 -32   14  -16;
 76                     -32 -32   8   8  64  -16 -16  -16   32;
 77                       2  14  14   2 -16  112 -16   16 -128;
 78                       8   8 -32 -32 -16  -16  64  -16   32;
 79                      14   2   2  14 -16   16 -16  112 -128;
 80                     -16 -16 -16 -16  32 -128  32 -128  256];
 81
 82  % 行列 Dhatxx の作成
 83  Dhatxx=assem_interval1(Dhatxx0,Nx,Ny);
 84
 85  % 行列 Dhatxy0 の作成
 86  Dhatxy0=intval(1)/36*[9  -3  -1   3  12   4   4 -12 -16;
 87                        3  -9  -3   1 -12  12  -4  -4  16;
 88                       -1   3   9  -3   4 -12  12   4 -16;
 89                       -3   1   3  -9  -4  -4 -12  12  16;
 90                      -12  12   4  -4   0 -16   0  16   0;
 91                        4 -12  12  -4 -16   0  16   0   0;
 92                        4  -4 -12  12   0  16   0 -16   0;
 93                       12  -4   4 -12  16   0 -16   0   0;
 94                      -16  16 -16  16   0   0   0   0   0];
 95
 96  % 行列 Dhatxy の作成
 97  Dhatxy=assem_interval1(Dhatxy0,Nx,Ny);
 98
 99  % 行列 Dhatyy0 の作成
100  Dhatyy0=hx/(90*hy)*[28  -7  -1   4   14   8    2 -32  -16;
101                      -7  28   4  -1   14 -32    2   8  -16;
102                      -1   4  28  -7    2 -32   14   8  -16;
103                       4  -1  -7  28    2   8   14 -32  -16;
104                      14  14   2   2  112 -16   16 -16 -128;
105                       8 -32 -32   8  -16  64  -16 -16   32;
106                       2   2  14  14   16 -16  112 -16 -128;
107                     -32   8   8 -32  -16 -16  -16  64   32;
108                     -16 -16 -16 -16 -128  32 -128  32 256];
109
110  % 行列 Dhatyy の作成
111  Dhatyy=assem_interval1(Dhatyy0,Nx,Ny);
112
113  % 行列 Dxx の作成
114  Dxx=delete_boundary(Dhatxx,Nx,Ny);
115
116  % 行列 Dxy の作成
```

```
117 Dxy=delete_boundary(Dhatxy,Nx,Ny);
118
119 % 行列 Dyy の作成
120 Dyy=delete_boundary(Dhatyy,Nx,Ny);
121
122 % 行列 Kx0 の作成
123 Kx0=hy/180*[-12   4  -1   3 -16   2   4  -6  -8;
124              -4  12  -3   1  16   6  -4  -2   8;
125               1  -3  12  -4  -4   6  16  -2   8;
126               3  -1   4 -12   4   2 -16  -6  -8;
127              16 -16   4  -4   0  -8   0   8   0;
128              -2   6   6  -2   8  48   8 -16  64;
129              -4   4 -16  16   0  -8   0   8   0;
130              -6   2   2  -6  -8  16  -8 -48 -64;
131               8  -8  -8   8   0 -64   0  64   0];
132
133 % 行列 Kx の作成
134 Kx=assem_interval2(Kx0,Nx,Ny);
135
136 % 行列 Ky0 の作成
137 Ky0=hx/180*[-12   3  -1  4   -6   4   2 -16  -8;
138               3 -12   4  -1  -6 -16   2   4  -8;
139               1  -4  12  -3  -2  16   6  -4   8;
140              -4   1  -3  12  -2  -4   6  16   8;
141              -6  -6   2   2 -48  -8  16  -8 -64;
142              -4  16 -16   4   8   0  -8   0   0;
143              -2  -2   6   6 -16   8  48   8  64;
144              16  -4   4 -16   8   0  -8   0   0;
145               8   8  -8  -8  64   0 -64   0   0];
146
147 % 行列 Ky の作成
148 Ky=assem_interval2(Ky0,Nx,Ny);
149
150 % 行列 Fhatxx0 の作成
151 Fhatxx0=hy/(6*hx)*[ 1 -1  0  0;
152                    -1  1  0  0;
153                     0  0  1 -1;
154                     0  0 -1  1;
155                     0  0  0  0;
156                    -2  2  2 -2;
157                     0  0  0  0;
158                     2 -2 -2  2;
159                     0  0  0  0];
160
161 % 行列 Fhatxx の作成
162 Fhatxx=assem_interval3(Fhatxx0,beta,Nx,Ny);
163
164 % 行列 Fhatxy0 の作成
165 Fhatxy0=intval(1)/36*[  5   1  -1  -5;
166                        -1  -5   5   1;
167                        -1  -5   5   1;
168                         5   1  -1  -5;
169                        -4   4  -4   4;
170                        -4 -20  20   4;
171                        -4   4  -4   4;
172                        20   4  -4 -20;
173                       -16  16 -16  16];
174
```

```
175 % 行列 Fhatxy の作成
176 Fhatxy=assem_interval3(Fhatxy0,beta,Nx,Ny);
177
178 % 行列 Fhatyx0 の作成
179 Fhatyx0=intval(1)/36*[  5  -5  -1   1;
180                         5  -5  -1   1;
181                        -1   1   5  -5;
182                        -1   1   5  -5;
183                        20 -20  -4   4;
184                        -4   4  -4   4;
185                        -4   4  20 -20;
186                        -4   4  -4   4;
187                       -16  16 -16  16];
188
189 % 行列 Fhatyx の作成
190 Fhatyx=assem_interval3(Fhatyx0,beta,Nx,Ny);
191
192 % 行列 Fhatyy0 の作成
193 Fhatyy0=hx/(6*hy)*[ 1  0  0 -1;
194                     0  1 -1  0;
195                     0 -1  1  0;
196                    -1  0  0  1;
197                     2  2 -2 -2;
198                     0  0  0  0;
199                    -2 -2  2  2;
200                     0  0  0  0;
201                     0  0  0  0];
202
203 % 行列 Fhatyy の作成
204 Fhatyy=assem_interval3(Fhatyy0,beta,Nx,Ny);
205
206 % 行列 Dtilde0 の作成
207 Dtilde0=1/(6*hx*hy)*[2*hx2+2*hy2    hx2-2*hy2     -hx2-hy2   -2*hx2+hy2;
208                        hx2-2*hy2  2*hx2+2*hy2   -2*hx2+hy2     -hx2-hy2;
209                         -hx2-hy2   -2*hx2+hy2  2*hx2+2*hy2    hx2-2*hy2;
210                       -2*hx2+hy2     -hx2-hy2    hx2-2*hy2  2*hx2+2*hy2];
211
212 % 行列 Dtilde の作成
213 Dtilde=assem_interval4(Dtilde0,beta,Nx,Ny);
214
215 % 行列 Ex0 の作成
216 Ex0=hy/36*[-5 1 0  0  4  2  0 -10  8;
217            -1 5 0  0 -4 10  0  -2 -8;
218             0 0 5 -1  0 10 -4  -2 -8;
219             0 0 1 -5  0  2  4 -10  8];
220
221 % 行列 Ex の作成
222 Ex=assem_interval5(Ex0,beta,Nx,Ny);
223
224 % 行列 Ey0 の作成
225 Ey0=hx/36*[-5  0 0 1 -10  0  2  4  8;
226             0 -5 1 0 -10  4  2  0  8;
227             0 -1 5 0  -2 -4 10  0 -8;
228            -1  0 0 5  -2  0 10 -4 -8];
229
230 % 行列 Ey の作成
231 Ey=assem_interval5(Ey0,beta,Nx,Ny);
232
```

```
233 %                                   %
234 % 基底関数からの行列作成 ここまで %
235 %                                   %
236
237 % 行列 D の作成
238 D=[         nu*D0  zeros(dim_phi);
239    zeros(dim_phi)          nu*D0];
240
241 % 行列 E の作成
242 E=[Ex Ey];
243
244 % 行列 G の作成
245 G=[ D            -E';
246    -E zeros(dim_psi)];
247
248 % 行列 invG の作成
249 invG=inv(G);
250
251 % 行列 Ga, Gb, Gstar の作成
252 Ga=invG(1:2*dim_phi,1:2*dim_phi);
253 Gb=invG(2*dim_phi+1:2*dim_phi+dim_psi,1:2*dim_phi);
254 Gstar=invG(2*dim_phi+1:2*dim_phi+dim_psi,2*dim_phi+1:2*dim_phi+dim_psi);
255
256 % 行列 invLhat の作成
257 invLhat=inv(Lhat);
258
259 % 行列 Mx, My の作成
260 Mx=Kx*invLhat;
261 My=Ky*invLhat;
262
263 % 行列 invL の作成
264 invL=inv(L);
265
266 % 行列 F の作成
267 F=[         invL  zeros(dim_phi);
268    zeros(dim_phi)          invL];
269
270 % 行列 Q1 の作成
271 tmpQ1=D0-Mx*Kx'-My*Ky';
272 Q1=[         tmpQ1 zeros(dim_phi);
273     zeros(dim_phi)         tmpQ1];
274
275 % 行列 A1 の作成
276 A1=Ga*Q1*Ga;
277
278 % 行列 Ehatxx, Ehatxy, Ehatyy の作成
279 Ehatxx=Mx*Dhatxx*Mx';
280 Ehatxy=Mx*Dhatxy*My';
281 Ehatyy=My*Dhatyy*My';
282
283 % 行列 E1 の作成
284 tmpE1=Ehatxx+Ehatxy+Ehatxy'+Ehatyy;
285 E1=[         tmpE1 zeros(dim_phi);
286     zeros(dim_phi)         tmpE1];
287
288 % 行列 E2 の作成
289 E2=[(Mx*Fhatxx+My*Fhatyx)*Gb;
290     (Mx*Fhatxy+My*Fhatyy)*Gb];
```

```
291
292 % 行列 E3 の作成
293 tmpE3=-(Kx*invLhat*Kx'+Ky*invLhat*Ky')*invL;
294 tmpE3_2=-(Kx*invLhat*Kx'-Ky*invLhat*Ky')*invL;
295 E3=[        tmpE3  zeros(dim_phi);
296     zeros(dim_phi)         tmpE3];
297
298 % 行列 Q3 の作成
299 Q3=[Dxx  Dxy;
300     Dxy' Dyy];
301
302 % 行列 A3 の作成
303 A3=Ga*Q3*Ga;
304
305 % ベクトル f の作成
306 f1tilde=intval(zeros(dim_phihat,1));
307 f2tilde=intval(zeros(dim_phihat,1));
308
309 for i=0:2*Nx
310     for j=0:2*Ny
311         n=i*(2*Ny+1)+(j+1);
312         x=i*hx/2;
313         y=j*hy/2;
314         f1tilde(n)=f1_interval(x,y);
315         f2tilde(n)=f2_interval(x,y);
316     end
317 end
318
319 f1hat=Lhat*f1tilde;
320 f2hat=Lhat*f2tilde;
321
322 count=1;
323 for i=1+(2*Ny+1):dim_phihat-(2*Ny+1)
324     if mod(i,2*Ny+1)~=1 & mod(i,2*Ny+1)~=0
325        nb_num(count)=i;
326        count=count+1;
327     end
328 end
329
330 f1=f1hat(nb_num,:);
331 f2=f2hat(nb_num,:);
332
333 f=[f1;f2];
334
335 % C(uh,ph) の計算
336 C0=intval(1)/(2*pi);
337 h=max(sup(hx),sup(hy));
338
339 % 行列 Ehatx0 の作成
340 Ehatx0=hy/36*[-1 -1  0  0 -4 -2  0 -2 -8;
341               1  1  0  0  4  2  0  2  8;
342               0  0  1  1  0  2  4  2  8;
343               0  0 -1 -1  0 -2 -4 -2 -8];
344 % 行列 Ehaty0 の作成
345 Ehaty0=hx/36*[-1  0  0 -1 -2  0 -2 -4 -8;
346               0 -1 -1  0 -2 -4 -2  0 -8;
347               0  1  1  0  2  4  2  0  8;
348               1  0  0  1  2  0  2  4  8];
```

```
349
350 % 行列 Ehatx の作成
351 Ehatx=assem_interval6(Ehatx0,beta,Nx,Ny);
352
353 % 行列 Ehaty の作成
354 Ehaty=assem_interval6(Ehaty0,beta,Nx,Ny);
355
356 % 行列 Ehat の作成
357 Ehat=Ehatx*f1tilde+Ehaty*f2tilde;
358
359 % 行列 Kxhat の作成
360 Kxhat=assem_interval1(Kx0,Nx,Ny);
361
362 % 行列 Kyhat の作成
363 Kyhat=assem_interval1(Ky0,Nx,Ny);
364
365 % 行列 E4 の作成
366 tmpE4=Mx*Kxhat+My*Kyhat;
367 E4=[tmpE4                        zeros(dim_phi,dim_phihat);
368     zeros(dim_phi,dim_phihat)                    tmpE4];
369
370 % C(uh,pf) の作成
371 ftilde=[f1tilde;f2tilde];
372
373 lap_uh_lap_uh=f'*Ga'*E1*Ga*f;
374 lap_uh_nabla_ph=f'*Ga*E2*f;
375 lap_uh_f=f'*Ga*E4*ftilde;
376 nabla_ph_f=f'*Gb'*Ehat;
377 nabla_ph2=f'*Gb'*Dtilde*Gb*f;
378 norm_f2=f1tilde'*Lhat*f1tilde+f2tilde'*Lhat*f2tilde;
379
380 apo_A2=nu^2*lap_uh_lap_uh-2*nu*lap_uh_nabla_ph+2*nu*lap_uh_f-2*nabla_ph_f+nabla_ph2+norm_f2;
381
382 div_uh_0=sqrt(f'*A3*f);
383 C_uh_ph=nu*sqrt(f'*A1*f)+C0*h*sqrt(apo_A2)+div_uh_0;
```

以下は aposteriori_interval.m から呼ばれる関数のプログラムである。

assem_interval6.m

```
 1 % assem_interval6.m
 2 % 要素係数行列から全体行列を作り出す (interval)
 3 % Ehatx, Ehaty に対応
 4
 5 function ret = assem_interval6(A0,beta,Nx,Ny)
 6
 7 nnode_x=2*Nx+1;
 8 nnode_y=2*Ny+1;
 9
10 size_A=nnode_x*nnode_y;
11
12 nnode_x1=Nx+1;
13 nnode_y1=Ny+1;
14
15 A=intval(zeros(nnode_x1*nnode_y1,nnode_x*nnode_y));
16
17 num=[0 2*nnode_y 2*nnode_y+2 2 nnode_y 2*nnode_y+1 nnode_y+2 1 nnode_y+1];
18 num1=[0 nnode_y1 nnode_y1+1 1];
```

```
19
20  for p=1:Nx
21      for q=1:Ny
22          i=2*p-2;
23          j=2*q-2;
24          % l : 要素 e_pq の局所節点番号 1 の Φ hat での全体節点番号
25          % m : 要素 e_pq の局所節点番号 1 の φ     での全体節点番号
26          l=i*nnode_y+j+1;
27          m=i*nnode_y1/2+j/2+1;
28          for s=1:4
29              for t=1:9
30                  A(m+num1(s),l+num(t))= A(m+num1(s),l+num(t))+A0(s,t);
31              end
32          end
33      end
34  end
35
36  for i=1:nnode_x1*nnode_y1
37      for j=1:nnode_x*nnode_y
38          A(i,j)=A(i,j)-beta(i)*A(nnode_x1*nnode_y1,j);
39      end
40  end
41
42  A(nnode_x1*nnode_y1,:)=[];
43  ret=A;
```

f1_interval.m

```
1  % 外力密度 f1_interval(x,y)
2  % interval 用
3
4  function ret = f1_interval(x,y)
5  ret=intval(50)*(-2*x+y+x*y);
```

f2_interval.m

```
1  % 外力密度 f2_interval(x,y)
2  % interval 用
3
4  function ret = f2_interval(x,y)
5  ret=intval(20)*(1-5*x*y);
```

9.3.3 事後誤差限界を求める実験用のプログラム

aposteriori_test_floating.m

```
1  % aposteriori_test_floating.m
2  % |u-uh|_1,|p-ph|_0,|u-uh|_0 の事後誤差限界を floating で求める
3  % 実験用のプログラム
4
5  clear
6  format long e
7
8  N=5;
9  Nx=N;
```

```
10  Ny=N;
11
12  nu=1;
13  width=1;
14  height=1;
15
16  hx=width/Nx;
17  hy=height/Ny;
18
19  fprintf('計算開始\n');
20  tic
21  [C_uh_ph div_uh_0]=aposteriori_floating(Nx,Ny,nu,width,height);
22  aposteriori_time=toc;
23  fprintf('計算終了\nC(uh,ph) を求めるのに %f 秒かかりました\n',aposteriori_time);
24
25  C_data=zeros(15,2);
26  C_data(2,1)=5.385150014363885e-001;  C_data(2,2)=1.817205268752533e+000;
27  C_data(3,1)=4.121952692375586e-001;  C_data(3,2)=1.390942523449535e+000;
28  C_data(4,1)=3.297341065882071e-001;  C_data(4,2)=1.112679413166317e+000;
29  C_data(5,1)=2.707422794093243e-001;  C_data(5,2)=9.136129825620294e-001;
30  C_data(6,1)=2.280909123179216e-001;  C_data(6,2)=7.696870217415104e-001;
31  C_data(7,1)=1.975824628601747e-001;  C_data(7,2)=6.667370297297524e-001;
32  C_data(8,1)=1.742917160008234e-001;  C_data(8,2)=5.881429928076554e-001;
33  C_data(9,1)=1.556665352829482e-001;  C_data(9,2)=5.252927909716553e-001;
34  C_data(10,1)=1.405510366710497e-001; C_data(10,2)=4.742859227431048e-001;
35  C_data(11,1)=1.281718527224788e-001; C_data(11,2)=4.325126792230596e-001;
36  C_data(12,1)=1.177518919172741e-001; C_data(12,2)=3.973507847077594e-001;
37  C_data(13,1)=1.088752731310815e-001; C_data(13,2)=3.673968588487742e-001;
38  C_data(14,1)=1.012369796334603e-001; C_data(14,2)=3.416216303943538e-001;
39  C_data(15,1)=9.459733611208575e-002; C_data(15,2)=3.192163210575709e-001;
40
41  K3_data=zeros(15,1);
42  K3_data(2)=8.416495753362069e-002;
43  K3_data(3)=7.099515407610255e-002;
44  K3_data(4)=5.985604991228765e-002;
45  K3_data(5)=5.100876307940924e-002;
46  K3_data(6)=4.405157359371238e-002;
47  K3_data(7)=3.862624912175199e-002;
48  K3_data(8)=3.430531393588975e-002;
49  K3_data(9)=3.081354668638923e-002;
50  K3_data(10)=2.794083993689709e-002;
51  K3_data(11)=2.554351111620361e-002;
52  K3_data(12)=2.351508996258608e-002;
53  K3_data(13)=2.177891489440772e-002;
54  K3_data(14)=2.027704048189094e-002;
55  K3_data(15)=1.896595841433703e-002;
56
57  C0=1/(2*pi);
58  h=max(hx,hy);
59  beta=1/sqrt(4+2*sqrt(2));
60
61  u_uh_1=sqrt(1/(nu^2)+1/(beta^2))*C_uh_ph;
62  p_ph_0=(1/beta+nu/(beta^2))*C_uh_ph;
63
64  C_uh_ph
65  u_uh_1
66  p_ph_0
67
```

```
68  u_uh_0=nu*C_data(N,1)*u_uh_1+C_data(N,2)*div_uh_0+K3_data(N)*p_ph_0
```

aposteriori_test_interval.m

```
 1  % aposteriori_test_interval.m
 2  % |u-uh|_1,|p-ph|_0,|u-uh|_0 の事後誤差限界を interval で求める
 3  % 実験用のプログラム
 4
 5  clear
 6  format long e
 7
 8  N=5;
 9  Nx=N;
10  Ny=N;
11
12  nu=1;
13  width=1;
14  height=1;
15
16  hx=intval(width)/Nx;
17  hy=intval(height)/Ny;
18
19  fprintf('計算開始\n');
20  tic
21  [C_uh_ph div_uh_0]=aposteriori_interval(Nx,Ny,nu,width,height);
22  aposteriori_time=toc;
23  fprintf('計算終了\nC(uh,ph) を求めるのに %f 秒かかりました\n',aposteriori_time);
24
25  C0=intval(1)/(2*pi);
26  h=max(sup(hx),sup(hy));
27  beta=intval(1)/sqrt(4+2*sqrt(2));
28  nu=intval(nu);
29
30  C_data=zeros(15,2);
31  C_data(5,1)=2.707422795256720e-001;   C_data(5,2)=9.136129829546416e-001;
32  C_data(10,1)=1.405510431595060e-001;  C_data(10,2)=4.742859446382364e-001;
33  C_data(15,1)=9.459740946044298e-002;  C_data(15,2)=3.192165685697495e-001;
34
35  K3_data=zeros(15,1);
36  K3_data(5)=5.100876308621370e-002;
37  K3_data(10)=2.794084060492480e-002;
38  K3_data(15)=1.896597079889749e-002;
39
40  u_uh_1=sqrt(1/(nu^2)+1/(beta^2))*C_uh_ph;
41  p_ph_0=(1/beta+nu/(beta^2))*C_uh_ph;
42
43  infsup(C_uh_ph)
44  infsup(u_uh_1)
45  infsup(p_ph_0)
46
47  u_uh_0=nu*C_data(N,1)*u_uh_1+C_data(N,2)*div_uh_0+K3_data(N)*p_ph_0;
48  infsup(u_uh_0)
```

A　Ωが滑らかな場合の Lemma 2.1 の証明

Lemma 2.1 を示すには次の Lemma A.1 を示せばよい。

Lemma A.1
$\Omega \subset \mathbf{R}^N$ を有界領域、その境界は滑らかとする。このとき、次の作用素

$$\begin{array}{ccc} T : V^{\perp} & \to & L^2_0(\Omega) \\ \cup & & \cup \\ v & \mapsto & \operatorname{div} v \end{array}$$

は同型となる。

証明
この Lemma とその証明は V. Girault, P. A. Raviart [18] p.33 Lemma 3.2 による。
$v \in H^1_0(\Omega)^2$ とする。Gauss の発散定理より、

$$\int_{\Omega} \operatorname{div} v \; dx = \int_{\partial\Omega} v \cdot n d\sigma = 0.$$

また、T は線形、連続であるので

$$T \in \mathscr{L}(H^1_0(\Omega)^2,\ L^2_0(\Omega))$$

となる。次に T が 1 対 1 かつ上への写像であることを示す。$H^1_0(\Omega)^2 = V \oplus V^{\perp}$ であるので、$\mathrm{Ker}(T)$=$\{0\}$ となる。よって T は 1 対 1 である。$q \in L^2_0(\Omega)$ とする。$\operatorname{div} v = q$ となるような $v \in H^1_0(\Omega)^2$ をさがす。Ω が有界であることより、Ω を境界が滑らかになるように拡張することができて、

$$\exists \theta \in H^2(\Omega) \quad \text{s.t.} \quad \triangle\theta = q \quad \text{in } \Omega$$

を得る。ここで、

$$v_1 = \operatorname{grad} \theta$$

とすると $v_1 \in H^1(\Omega)^2$ である。このとき、

$$\operatorname{div} v_1 = \triangle\theta = q$$

でありさらに γ_0 をトレース作用素とすると、$\gamma_0 v_1 \in H^{\frac{1}{2}}(\partial\Omega)^2$ に対して

$$\int_{\partial\Omega} \gamma_0 v d\sigma = \int_{\partial\Omega} v_1 \cdot n d\sigma = \int_{\Omega} \operatorname{div} v_1 dx = \int_{\Omega} q \; dx = 0$$

である。ここで次の定理を用いる。

Theorem A.1
$\Omega \subset \mathbf{R}^N$ を有界領域、その境界は滑らかとする。また、$g \in H^{\frac{1}{2}}(\partial\Omega)^2,\ \displaystyle\int_{\partial\Omega} g \cdot n \; d\sigma = 0$ とする。このとき、

$$\exists u \in H^1(\Omega)^2 \quad \text{s.t.} \quad \operatorname{div} u = 0,\ u = g \quad \text{on } \partial\Omega.$$

この定理は V. Girault, P. A. Raviart [18] p.32 Theorem 3.5 によるものである。証明はあとで行う。この定理より、

$$\exists w_1 \in H^1(\Omega)^2 \quad \text{s.t.} \quad \operatorname{div} w_1 = 0,\ \gamma_0 w_1 = \gamma_0 v_1.$$

$v^* = v_1 - w_1$ とすると、

$$v^* \in H^1(\Omega)^2 \quad \text{かつ} \quad \operatorname{div} v^* = q.$$

ここで、

$$v^* = w + v, \qquad w \in V, \quad v \in V^\perp$$

とすると、

$$q = \operatorname{div} v^* = \operatorname{div} v$$

を得る。よって上への写像であることも示せた。div^{-1} は値域定理より連続である。□

Theorem A.1 の証明

この Theorem の証明は V. Girault, P. A. Raviart [18] p.32 Theorem 3.5 による。

(1) $g \cdot n = 0 \quad \text{on} \quad \partial\Omega$ の場合を示す。

以下の場合を示せば十分である。

$$\text{Find} \quad \psi \in H^2(\Omega) \quad \text{s.t.} \quad \frac{\partial \psi}{\partial \tau} = 0 \quad \text{on } \partial\Omega, \quad \frac{\partial \psi}{\partial n} = g \cdot \tau \quad \text{on } \partial\Omega. \tag{A.1}$$

ここで、$n = (n_1,\ n_2)$ は境界 $\partial\Omega$ 上の外向き単位法線ベクトルである。また、τ は n を 90 度回転させたベクトル、つまり

$$\tau = \begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix} = \begin{pmatrix} \cos(\pi/2) & -\sin(\pi/2) \\ \sin(\pi/2) & \cos(\pi/2) \end{pmatrix} \begin{pmatrix} n_1 \\ n_2 \end{pmatrix} = \begin{pmatrix} -n_2 \\ n_1 \end{pmatrix}$$

である。(A.1) の場合を示せば十分であることを以下に示す。

$$u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} -\dfrac{\partial \psi}{\partial y} \\ \dfrac{\partial \psi}{\partial x} \end{pmatrix}$$

とすると $\psi \in H^2(\Omega)$ より $u \in H^1(\Omega)^2$ で、

$$\operatorname{div} u = -\frac{\partial^2 \psi}{\partial x \partial y} + \frac{\partial^2 \psi}{\partial x \partial y} = 0$$

である。また、

$$u \cdot n = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \cdot \begin{pmatrix} n_1 \\ n_2 \end{pmatrix} = \begin{pmatrix} -\dfrac{\partial \psi}{\partial y} \\ \dfrac{\partial \psi}{\partial x} \end{pmatrix} \cdot \begin{pmatrix} \tau_2 \\ -\tau_1 \end{pmatrix} = -\frac{\partial \psi}{\partial \tau} = 0 = g \cdot n \quad \text{on } \partial\Omega,$$

$$u \cdot \tau = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \cdot \begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix} = \begin{pmatrix} -\dfrac{\partial \psi}{\partial y} \\ \dfrac{\partial \psi}{\partial x} \end{pmatrix} \cdot \begin{pmatrix} -n_2 \\ n_1 \end{pmatrix} = \frac{\partial \psi}{\partial n} = g \cdot \tau \quad \text{on } \partial\Omega.$$

よって、u は求める関数となる。ここで、次の定理を用いる。

Theorem A.2
$\Omega \subset \mathbf{R}^N$ を有界領域、その境界は滑らかとする。また $m \geq 2$ を整数とする。このとき写像

$$\begin{array}{ccc} H^m(\Omega) & \to & H^{m-\frac{1}{2}}(\partial\Omega) \times H^{m-\frac{3}{2}}(\partial\Omega) \\ \cup & & \cup \\ u & \mapsto & \left\{ \gamma_0 u,\ \dfrac{\partial u}{\partial n} \right\} \end{array}$$

は上への写像である。

この Theorem は J.L.Lions, E.Magenes [21] p.39 Theorem 8.3 と V. Girault, P. A. Raviart [17] p.5 Theorem 1.5 に載っている。前者には証明が載っているが、後者には証明が載っていない。

この定理より、

$$\exists \psi \in H^2(\Omega) \quad \text{s.t.} \quad \psi = 0 \quad \text{on } \partial\Omega, \quad \frac{\partial \psi}{\partial n} = g \cdot \tau \quad \text{on } \partial\Omega.$$

この ψ は (A.1) を満たす。

(2) $g \cdot n \neq 0$ on $\partial\Omega$ の場合を示す。

この時、$g \cdot n \notin H^{\frac{3}{2}}(\partial\Omega)$ より Theorem A.2 を直接に使えない。したがって次の Neumann 問題を考える。

$$\begin{cases} \triangle p = 0 & \text{in } \Omega, \\ \dfrac{\partial p}{\partial n} = g \cdot n & \text{on } \partial\Omega. \end{cases} \tag{A.2}$$

ここで、

$$\int_{\partial\Omega} g \cdot n d\sigma = 0$$

である。(A.2) は定数倍を除けば、$H^1(\Omega)$ で一意解 $\dot{p}$ を持つ。$g \cdot n \in H^{\frac{1}{2}}(\partial\Omega)$ より $\partial\Omega$ は十分に滑らかならば、$\dot{p} \in H^2(\Omega)$ となる。

(1) での結果から、

$$\exists w \in H^1(\Omega)^2 \quad \text{s.t.} \quad \operatorname{div} w = 0 \quad \text{on } \partial\Omega, \quad w = g - \gamma_0(\operatorname{grad} p) \quad \text{on } \partial\Omega.$$

$u = w + \operatorname{grad} p$ とすれば求める u を得る。□

謝辞

この論文を作成するにあたり、丁寧な指導と適切な助言をしてくださった桂田祐史助教授に深い感謝の意を表します。また昨年ご指導を頂きました森本浩子教授に厚く御礼を申し上げます。

参考文献

[1] Mitsuhiro T. Nakao, Nobito Yamamoto, Yoshitaka Watanabe, A posteriori and constructive a priori error bounds for finite element solutions of the Stokes equations, Journal of Computaitional and Applied Mathematics 91 (1998) 137-158.

[2] 中尾充宏、山本野人、渡部善隆、 Stokes 方程式の有限要素解に対する a posteriori 誤差評価、京都大学数理解析研究所講究録、928 (1995), pp.20–31.

[3] 中尾充宏、山本野人、渡部善隆、 Stokes 方程式の有限要素解に対する a priori 誤差評価、京都大学数理解析研究所講究録、944 (1996), pp.41–49.

[4] 中尾充宏、山本野人、渡部善隆、 Stokes 方程式の有限要素解に対する a posterioti 誤差評価、応用数学合同研究集会 (1996).

[5] 渡部善隆、山本野人、中尾充宏、Navier-Stokes 方程式の解の数値的検証法について、京都大学数理解析研究所講究録 (1998), pp.100–105.

[6] Watanabe Yoshitaka, Yamamoto Nobito and Nakao T.Mitsuhiro, A Numerical Verification Method of Solutions for the Navier Stokes Equations, Reliable Computing 5 (1999), pp.347–357.

[7] C.O.Horgan, L.E.Payne, On Inequalities of Korn, Friedrichs and Babuška-Aziz, Arch. Rat. Mech. Anal. 82 (1983) 165-179.

[8] Nobito Yamamoto, Mitsuhiro T.Nakao, Numerical verification of solutions for elliptic equations in nonconvex polygonal domains, Numer. Math. 65 (1993) 503-521.

[9] Mistuhiro T. Nakao, Nobito Yamamoto, Seiji Kimura, On the Best Constant in the Error Bound for the H_0^1-Projection into Piecewise Polynomial Spaces, Journal of Approximation Theory 93, (1998) 491-500.

[10] 渡部 善隆、山本 野人、中尾 充宏、一般化固有値問題の精度保証付き計算とその応用、日本応用数理学会論文誌 Vol. 9, No.3, (1999), pp 135〜150.

[11] 塩谷 光晴、定常 Stokes 方程式の解の精度保証付き数値計算、明治大学修士学位請求論文 (2002).

[12] 菊地 文雄、有限要素法の数理、培風館 (1994).

[13] 菊地 文雄、有限要素法概説 [新訂版]、サイエンス社 (1999).

[14] 土屋 卓也、東京大学大学院数理科学研究科「応用数学特別講義 II」講義ノート (2003). http://daisy.math.sci.ehime-u.ac.jp/users/tsuchiya/math/fem/index.html

[15] 桂田 祐史、一般化固有値問題 (2004). http://www.math.meiji.ac.jp/~mk/labo/text/generalized-eigenvalue-problem.pdf

[16] 中尾 充宏、山本 野人、精度保証付き数値計算、日本評論社 (1998).

[17] Vivette Girault, Pierre-Arnaud Raviart, Finite Element Methods for Navier-Stokes Equations, Theory and Algorithms, Springer-Verlag (1986).

[18] V.Girault, P.-A.Raviart, Finite Element Approximation of the Navier-Stokes Equations, Springer-Verlag (1981).

[19] A.K.Aziz, The Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations, Academic Press (1972).

[20] Roger Temam, Navier-Stokes Equations, Theory and Numerical Analysis, Third (revised) edition, North-Holland, Amsterdam (1984).

[21] J.L.Lions, E.Magenes, translated from the French by P.Kenneth, Non-Homogeneous Boundary Value Problems and Applications Volume I, Springer-Verlag (1972).

[22] ハイム・ブレジス (藤田 宏 監訳、小西 芳雄 訳)、関数解析、産業図書 (1988).

[23] 藤田 宏、黒田 成俊、伊藤 清三、関数解析、岩波基礎数学選書 (1991).