半線形楕円型偏微分方程式に対する精度保証付き数値計算

明治大学大学院
理工学研究科 基礎理工学専攻 数学系

鈴木康大

指導教員　桂田祐史 助教授

目 次

1 はじめに **4**

2 精度保証付き数値計算とは **5**
2.1 計算誤差について 5
2.2 浮動小数点数 5
2.3 浮動小数点数への丸め 6
2.4 区間演算 7
2.5 機械区間演算 7

3 有限要素法とその a priori 誤差評価 **9**
3.1 Poisson 方程式の有限要素解に対する誤差評価 9
3.2 C_0, C_1 の具体的な値 11

4 線形楕円型偏微分方程式に対する精度保証付き数値計算 **12**
4.1 考える問題とその有限要素解 12
4.2 不動点形式 12
4.3 候補者集合 13
4.4 検証条件 14
4.5 区間演算 15
4.6 具体的な数値例 16
4.6.1 有限要素法 16
4.6.2 区間演算 18
4.6.3 具体的なアルゴリズム 19
4.6.4 数値計算結果 20
4.6.5 非強圧的線形楕円型偏微分方程式に対する数値実験 23

5 半線形楕円型偏微分方程式に対する精度保証付き数値計算 **26**
5.1 残差誤差と残差方程式 26
5.2 $\|\nabla v_0\|$ の a priori 評価 27
5.3 $\|\nabla v_0\|$ の a posteriori 評価 28
5.4 擬似 Newton 法 29
5.5 候補者集合 30
5.6 検証条件 30
5.7 具体的な数値例 31
5.7.1 有限要素法 31
5.7.2 $\|\nabla v_h\|$ の評価 33
5.7.3 $\|\nabla v_0\|$ の a posteriori 評価 34
5.7.4 $T(w)$ の具体的な評価 37
5.7.5 $|\vec{g}_h|$ の評価 37
5.7.6 無限次元部分の評価 38
5.7.7 具体的な検証条件とアルゴリズム 39
5.7.8 数値計算結果 40
5.7.9 擬似 Newton 法による非強圧的線形楕円型偏微分方程式の解の検証 42

A 線形楕円型偏微分方程式に対する精度保証付き数値計算のプログラム **43**

B 半線形楕円型偏微分方程式に対する精度保証付き数値計算のプログラム **48**

1　はじめに

精度保証付き数値計算とは数値計算の結果を「数学的に」正しいと保証するものである。今までの数値計算では、計算結果が真の解を近似しているとしても、その近似解のどのくらいの近傍に真の解が存在しているか、ということに関しては証明することが困難であった。

しかし、コンピュータの性能が向上したことや、数値計算理論の発展に伴って、この十数年間において、数値計算結果を数学的に保証しようとする動きが活発になっている。特に、連立一次方程式、常微分方程式に関してはかなりの成果が上がってきている (大石 [11])。しかし、偏微分方程式については、その理論の複雑さにより未だ発展途上である。

その偏微分方程式の中でも、楕円型偏微分方程式などの精度保証付き数値計算については、九州大学の中尾充宏教授が多くの論文を発表なされている (中尾 [8], Nakao [9], Yamamoto, Nakao [16])。その基本的な理論は、対象とする方程式を不動点方程式に変換し、真の解を含む可能性のある関数の集合を適切に設定したうえで、Schauder の不動点定理を適用することで、解の存在範囲を数学的に得ることを目指す。さらに、こうして得られた数学的な解の存在範囲を区間演算という数値計算の理論を用いて、数値的な解の保証区間を得ようとするものである。

この理論では、数学的考察のみならず、実際に数値計算を行うことで解の存在証明ができる。したがって、数値計算の部分も追試が必要である。しかし、中尾教授の論文には、追試が可能なだけの情報が具体的に書かれていなかった。

そこで、ここでは、誰にでも再現可能な数値計算の手順を示すということに目標を置くこととした。

本論文では、楕円型偏微分方程式のうち、Dirichlet 境界条件のついた Poisson 方程式 $-\Delta u = f(x, u, \nabla u)$ の右辺が $u, \nabla u$ について線形な方程式

$$\begin{cases} -\Delta u = \pi u + (2\pi - 1)\sin \pi x \sin \pi y, & \text{in } \Omega = (0,1) \times (0,1) \\ u = 0, & \text{on } \partial\Omega \end{cases}$$

と、非強圧的な線形方程式

$$\begin{cases} -\Delta u = 20xyu + (2\pi - 1)\sin \pi x \sin \pi y, & \text{in } \Omega = (0,1) \times (0,1) \\ u = 0, & \text{on } \partial\Omega \end{cases}$$

と、さらに、$f(x, u, \nabla u)$ が u についての高次多項式である半線形楕円型偏微分方程式

$$\begin{cases} -\Delta u = u^2, & \text{in } \Omega = (0,1) \times (0,1) \\ u = 0, & \text{on } \partial\Omega \end{cases}$$

について扱うこととする。

この論文の概略は次のようなものである。まず、第 2 節で、数値実験をするにあたって必要な浮動小数点数システム、数値の丸めなど計算機に関わることや、区間演算の理論を説明する。

次に第 3 節で、この論文で扱う楕円型偏微分方程式の精度保証付き数値計算の基礎となる Poisson 方程式の有限要素解とその構成的誤差評価について簡単ではあるが述べる。

第 4 節において、線形楕円型偏微分方程式に対する精度保証付き数値計算の理論を紹介し、実際に数値実験を行い、その理論を確かめる。

最後に第 5 節において、残差誤差の a posteriori 評価と残差方程式に対して用いる擬似 Newton 法とを組み合わせた、半線形楕円型偏微分方程式に対する精度保証の手順を述べ、この方法に従い数値実験を行う。

2　精度保証付き数値計算とは

2.1　計算誤差について

数値計算の結果はすべてが正しいとは限らない。浮動小数点数演算における数値的結果には誤差が伴うこととなる。その誤差の原因としては

1. モデル化誤差
2. 離散化誤差
3. 丸め誤差

がある。モデルの誤差はどこまで現象を記述したモデルを使っているかで決まる。離散化誤差は微分方程式の微分を数値的に解く場合には、微分方程式を差分方程式で近似するときに生じる誤差である。丸め誤差は計算機の中で扱う数値が無限桁ではなく、有限桁であることにより生じる誤差である。

2.2　浮動小数点数

浮動小数点数システムについては、IEEE 標準 754 (IEEE754 と略記する) がパソコンやワークステーションなどで、標準的に用いられている。IEEE754 によって規定される 2 進浮動小数点数としては 4 つのタイプの数が用意されている。それは規格化 2 進浮動小数点数、零、非規格化 2 進浮動小数点数、NaN (非数) である。

2 進規格化浮動小数点数 2 進規格化浮動小数点数とは、N をある正整数として、

$$a = \pm\left(\frac{1}{2} + \frac{d_2}{2^2} + \frac{d_3}{2^3} + \cdots + \frac{d_N}{2^N}\right) \times 2^e, \quad d_i = 0 \text{ or } 1$$

と書ける数をいう。$e_{\min}$ を負の整数、 $e_{\max}$ を正の整数として、e は $e_{\min} \le e \le e_{\max}$ となる整数である。

$$m = \pm\frac{1}{2} + \frac{d_2}{2^2} + \frac{d_3}{2^3} + \cdots + \frac{d_N}{2^N}$$

を符号付き仮数といい、e を指数という。指数 e も 2 進数で表される。

2 進浮動小数点数システムにおいて表される数の絶対値の最大値は

$$x_{\max} = \left(\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \cdots + \frac{1}{2^N}\right) 2^{e_{\max}}$$

であり、その最小値は

$$x_{\min} = \frac{1}{2} 2^{e_{\min}}$$

である。$|x| > x_{\max}$ のときにオーバフローが生じたという。

零 零は規格化されて

$$+\left(\frac{0}{2} + \frac{0}{2^2} + \frac{0}{2^3} + \cdots + \frac{0}{2^N}\right) 2^{e_{\min}}$$

と表される。

非規格化 2 進浮動小数点数 IEEE754 では、浮動小数点数は指数部が $e_{\min}$ となったとき、仮数部の最初の桁が 1 より小さい数を表すために、デフォルトで最初の桁を 1 とすることをやめ、ここが 0 となる数をおくことを許す規格となっている。これを非規格化数という。このような非規格化浮動小数点数の一番小さい正の数は

$$2^{-1074} = 4.940\ 656\ 458\ 412\ 465 \times 10^{-323}$$

である。これ以下の数になると、アンダフローが生じたという。

NaN この他に、IEEE754 では、次のような特別な数が用意されている。

1. NaN (Not a Number) は $\sqrt{-2}$, ∞/∞, $+\infty + (-\infty)$ など不当な演算の結果として得られる。
2. $\pm\infty$ はオーバフローの結果やゼロで割った結果として得られる。
3. ± 0 はアンダフローか $\pm\infty$ での割り算の結果として得られる。

これより、以上の 4 つのタイプの数のうち NaN を除いたもの、つまり、規格化浮動小数点数、零、非規格化浮動小数点数の集合を $\mathbf{F}$ で表す。

2.3 浮動小数点数への丸め

IEEE754 では、浮動小数点数の集合 $\mathbf{F}$ 上での演算は丸めを用いて定義されている。また、次の四つの丸めのモードが指定できるように、コンピュータが設計されていることを要請している。c を実数 ($c \in \mathbf{R}$) としたとき、

1. 上向きの丸め　　c 以上の浮動小数点数の中でもっとも小さい数に丸める。これを $\bigtriangledown : \mathbf{R} \to \mathbf{F}$ と表す。
2. 下向きの丸め　　c 以下の浮動小数点数の中でもっとも大きい数に丸める。これを $\bigtriangleup : \mathbf{R} \to \mathbf{F}$ と表す。
3. 最近点への丸め　　c に最も近い浮動小数点数に丸める。これを $\Box : \mathbf{R} \to \mathbf{F}$ と表す。もし、このような点が 2 点ある場合には仮数部の最後のビットが偶数である浮動小数点数に丸める。
4. 切り捨て　　絶対値が c 以下の浮動小数点数の中で c に最も近いものに丸める。

丸めの演算を写像として、$\bigcirc : \mathbf{R} \to \mathbf{F}$ と書く。すなわち、$\bigcirc$ は $\bigtriangledown, \bigtriangleup, \Box$ のいずれかと考える。IEEE754 では丸めの演算は次の条件を満たすことが要請されている。

$$\begin{cases} \bigcirc x = x & (\forall x \in \mathbf{F}) \\ x \le y \Rightarrow \bigcirc x \le \bigcirc y & (\forall x, y \in \mathbf{R}) \end{cases}$$

また、$x \in \mathbf{F}$ のとき、符号を変えることにより、$-x$ や $|x|$ が得られるので、これらは正確に計算される。

IEEE754 では次の性質が成立することも要請されている。

$$\begin{cases} \Box(-x) = -\Box x & (\forall x \in \mathbf{R}) \\ \bigtriangledown(-x) = -\bigtriangledown x & (\forall x \in \mathbf{R}) \\ \bigtriangleup(-x) = -\bigtriangleup x & (\forall x \in \mathbf{R}) \end{cases}$$

IEEE754 では浮動小数点数の演算 ($\mathbf{F}$ 上での四則演算) は丸めとの関係により、次のように定義されている。$\cdot \in \{+, -, \times, /\}$、$\bigcirc \in \{\triangle, \bigtriangledown, \square\}$ のとき

$$x \odot y = \bigcirc(x \cdot y), \quad (\forall x, y \in \mathrm{R})$$

この式は、左辺の浮動小数点数の四則演算の結果 $x \odot y$ は、右辺の数学的に正しい (実数としての) 四則演算の結果 $x \cdot y$ を指定された丸めを行って得られた数 $\bigcirc(x \cdot y)$ に一致するように計算する規格を表している。

2.4 区間演算

ここでは、まず、実数体 $\mathbf{R}$ 上での区間の概念を述べる。したがって、四則演算は正確にできるものとする。実数の区間とは次のような閉区間である。

$$[a, b] = \{x \in \mathbf{R} \ ; \ a \leq x \leq b\}$$

ただし、$a \leq b, \ a, b \in \mathbf{R}$ である。

2 つの区間 $X = [a, b]$, $Y = [c, d]$ に対し、その 2 つの区間の四則演算を次で定義する。

$$X \circ Y \equiv \{x \circ y \ ; \ x \in X, y \in Y\} \tag{2.1}$$

ただし、$\circ \in \{+, -, \times, /\}$ である。$\circ$ が $/$ の場合には、Y は 0 を含まないものとする。

(2.1) で定義される実数の集合はまた区間になり、区間全体の集合を IR と書くとき、IR は演算 $\circ$ に関して閉じている。つまり、$X, Y \in IR \Rightarrow X \circ Y \in IR$ が成り立つ。

実際には定義 (2.1) は次の形に表現できる。

$$\begin{aligned}
X + Y &= [a + c, b + d] \\
X - Y &= [a - d, b - c] \\
X \times Y &= [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}] \\
X/Y &= [a, b] \times \left[\frac{1}{d}, \frac{1}{c}\right]
\end{aligned}$$

区間演算については、加法と乗法の交換則、結合則が成立するが、分配則は成立しない。その代わり、次の劣分配則が成り立つ。

$$A \times (B + C) \subseteq A \times B + A \times C, \qquad A, B, C \in IR \tag{2.2}$$

(2.2) で等号は、例えば、区間 B, C が同じ符号を持つときか、$A = [a, a]$ (点区間) であるときに成立する。

さらに、区間 X の幅を $d(X) \equiv b - a$、 絶対値を $|X| \equiv \max\{|a|, |b|\}$、中点を $m(X) \equiv \frac{a+b}{2}$ と定義する。また、区間ベクトル、区間行列をそれぞれ、その成分および要素が区間であるようなベクトル、行列として定める。区間ベクトル、区間行列に対しても (2.2) の劣分配則は成り立つが、乗法の結合則は成り立たないことに注意する。

2.5 機械区間演算

前小節で説明した区間演算では、区間の両端の数は実数であるとして、厳密な実数演算に基づいて区間演算が定義できた。しかし、浮動小数点数システム上で数値計算する際には、厳密には実行できない。そこで、区間演算を浮動小数点数システム $\mathbf{F}$ 上に展開する方法を述べる。

$$IF = \{[a, b] \ ; \ a, b \in \mathbf{F}, a \leq b\}$$

とする。区間 $[a,b] \in IR$ を区間 $[c,d] \in IF$ に丸める区間の丸め $\diamondsuit : IR \to IF$ を

$$X \subset \diamondsuit X$$

および、次を満たすように定義する。

$$\begin{array}{ll} \diamondsuit X = X & (\forall X \in IR) \\ X \subset Y \Rightarrow \diamondsuit X \subset \diamondsuit Y & (\forall X, Y \in IR) \\ \diamondsuit(-X) = -\diamondsuit X & (\forall X \in IR) \end{array}$$

IEEE754 上の浮動少数点数システムにおいては $X = [a,b]$ として、

$$\diamondsuit X = [\bigtriangledown a, \bigtriangleup b]$$

と定義すれば、上の条件は満たされる。そして、機械区間演算の四則演算を

$$X \hat{\diamondsuit} Y = \diamondsuit(X \circ Y)$$

で定義する。ただし、$\circ \in \{+, -, \times, /\}$ である。

これを用いると浮動小数点数の四則演算結果は基本的に厳密に把握できる。例えば、

$$\begin{array}{rcl} 2 \div 3 \times \sqrt{2} & \in & [0.666, 0.667] \times [1.414, 1.415] \\ & \subset & [0.941, 0.944] \end{array}$$

というように。

しかし、このような素朴な区間演算では、解が含まれる区間の幅は演算回数を重ねるごとに累積していき、最終結果としてはほとんど有意な情報をもたらさないこともある。このことがあり、区間演算はあまり積極的な評価を受けてこなかったが、現在では、この点を克服する効率的区間演算の実現技術が登場し、その評価も大きく変わりつつある。

実際の精度保証を実現する上で用いられている原理的手法は次の 3 つに要約される。

1. ベクトル内積の完全精度による計算
2. 区間演算と不動点定理の適用による計算結果の数学的に完全な保証
3. 反復残差改良法

ここでは、主に 2. を用いた偏微分方程式の精度保証について述べる。1. については 杉浦 [15] を、3. については Rump [12], Rump [13] を参照されたい。

3　有限要素法とその a priori 誤差評価

a priori 誤差評価とは、ひとつの近似スキームによって近似解を計算する際に、前もって見込まれる誤差限界で、事前誤差評価とも呼ばれる。これに対し、近似解を計算した後で、その近似解および問題自体の特徴を生かして見積もる誤差を a posteriori 誤差評価といい、事後誤差評価とも呼ばれる。

ここでは、有限要素法とその誤差評価については簡単に説明するのみなので、有限要素法一般については、菊地 [5] を参照されたい。

3.1　Poisson 方程式の有限要素解に対する誤差評価

ここでは、Poisson 方程式

$$\begin{cases} -\Delta\xi(x) = g(x), & x = (x_1, x_2) \in \ \Omega \subset \mathbf{R}^2 \\ \xi(x) = 0, & x \in \ \partial\Omega \end{cases} \tag{3.1}$$

の有限要素近似解の誤差評価について述べる。

ここで、Ω は $\mathbf{R}^2$ 上の有界凸領域でその境界 $\partial\Omega$ は区分的に滑らかであるものとし、$\Delta\xi \equiv \frac{\partial^2\xi}{\partial x_1^2} + \frac{\partial^2\xi}{\partial x_2^2}$ である。

(3.1) の弱解とは、次式を満たす $\xi \in H_0^1(\Omega)$ として定義される。

$$(\nabla\xi, \nabla\psi)_{L^2} = (g, \psi)_{L^2}, \quad \forall\psi \in H_0^1(\Omega) \tag{3.2}$$

ここで、$(\cdot,\cdot)_{L^2}$ は Ω 上の L^2 内積であり、$\|\cdot\|$ を $L^2 \equiv L^2(\Omega)$ のノルムであるとする。また、$\nabla\xi = \left(\frac{\partial\xi}{\partial x_1}, \frac{\partial\xi}{\partial x_2}\right)$ である。

$H^1(\Omega)$ を

$$H^1(\Omega) \equiv \left\{\psi \in L^2(\Omega) \ ; \ \nabla\psi \in L^2(\Omega)\right\}$$

で定義し、$H_0^1 \equiv H_0^1(\Omega)$ を $C_0^\infty(\Omega)$ の $H^1(\Omega)$ における閉包として定義する。

H_0^1 の内積は、$\langle\phi,\psi\rangle \equiv (\nabla\phi, \nabla\psi)_{L^2}$ で定義され、そのノルムは $\|\psi\|_{H_0^1} = \|\nabla\psi\| = \sqrt{(\nabla\psi, \nabla\psi)}$ である。

$g \in L^2$ に対して、(3.2) は一意解 $\xi \in H^2(\Omega) \cap H_0^1$ を持ち、

$$|\xi|_{H^2} \le C_1\|g\| \tag{3.3}$$

となる正定数 C_1 が存在する (Grisvard [3])。

ここで、$|\cdot|_{H^2} \equiv |\cdot|_{H^2(\Omega)}$ は $H^2 \equiv H^2(\Omega)$ 上のセミノルムで、

$$|\xi|_{H^2}^2 \equiv \sum_{i,j=1}^{2} \left\| \frac{\partial^2\xi}{\partial x_i \partial x_j} \right\|^2$$

で定義される。

また、$H^m(\Omega)$ とは次で定義されるような関数空間である。

$$H^m(\Omega) \equiv \left\{\psi \in L^2 \ ; \ D^\alpha\psi \in L^2, \ |\alpha| \le m\right\}$$

ただし、$\alpha = \{\alpha_1, \alpha_2\}$, $0 \le \alpha_i \in \mathbf{Z}$ $(i = 1, 2)$ で、$|\alpha| = \alpha_1 + \alpha_2$、さらに、$D^\alpha$ は

$$D^\alpha \equiv \left(\frac{\partial}{\partial x_1}\right)^{\alpha_1} \left(\frac{\partial}{\partial x_2}\right)^{\alpha_2}$$

で表される超関数の意味の微分演算子を意味する。

まず、H_0^1 の有限次元部分空間 S_h で次の仮定を満たすものをとる。

$$\inf_{\phi \in S_h} \|\nabla (\xi - \phi)\| \leq C_0 h |\xi|_{H^2}, \quad \forall \xi \in H^2 \cap H_0^1 \tag{3.4}$$

ここで、C_0 は正定数である。このような仮定 (3.4) は、一般の多くの有限要素空間で成り立つことが知られている。また、C_0 の値が定数として分かっていたり、数値計算によって算定可能であるような S_h の例は多い (Schultz [14], 中尾 [10])。

(3.1) に対する有限要素近似解とは、次式を満たす $\xi_h \in S_h$ のことであると定義される。

$$(\nabla \xi_h, \nabla \phi)_{L^2} = (g, \psi)_{L^2}, \quad \forall \phi \in S_h$$

また、H_0^1-projection $P_h : H_0^1 \to S_h$ を、次のように定義する。

$$(\nabla(\psi - P_h \psi), \nabla \phi)_{L^2} = 0, \quad \forall \phi \in S_h \tag{3.5}$$

(3.4)、(3.5) を用いて、次の a priori H_0^1 誤差評価を得る。

$$\|\nabla (\xi - P_h \xi)\| \leq C_0 h |\xi|_{H^2} \tag{3.6}$$

なんとなれば、P_h は H_0^1 での直交射影 (H_0^1 ノルムの意味での最良近似) であるから、

$$\|\nabla (\xi - P_h \xi)\| = \inf_{\phi \in S_h} \|\nabla (\xi - \phi)\|$$

が成り立つ。

また、次のような a priori L^2 誤差評価を得ることができる。

$$\|\xi - P_h \xi\| \leq C_0^2 C_1 h^2 |\xi|_{H^2} \tag{3.7}$$

証明

$e = \xi - P_h \xi$ とおいて、次の方程式の弱解 η を考える。

$$\begin{cases} -\Delta \eta = e & \text{in } \Omega \\ \eta = 0 & \text{on } \partial\Omega \end{cases}$$

この方程式の弱形式は次式で与えられる。

$$(\nabla \eta, \nabla \psi) = (e, \psi), \quad \forall \psi \in H_0^1$$

(3.5) において、$\psi = \xi,\ \phi = P_h \eta$ とすると、$(\nabla e, \nabla P_h \eta) = 0$. したがって、

$$\begin{aligned} \|\xi - P_h \xi\|^2 &= (e, e)_{L^2} = (\nabla e, \nabla \eta)_{L^2} = (\nabla e, \nabla (\eta - P_h \eta))_{L^2} \\ &\leq \|\nabla e\| \|\nabla (\eta - P_h \eta)\| \\ &\leq \|\nabla e\| C_0 h |\eta|_{H^2} \qquad (\because\ (3.6)) \\ &\leq \|\nabla e\| C_0 h C_1 \|e\| \qquad (\because\ (3.3)) \\ &\leq C_0^2 C_1 h^2 |\xi|_{H^2} \|e\| \qquad (\because\ (3.6)) \end{aligned}$$

が成り立つ。これより、(3.7) を得る。 ∎

このように a priori L^2 誤差評価を得る手法は、 Aubin-Nitsche の技巧と呼ばれる。つまり、Aubin-Nitsche の技巧とは、a priori L^2 誤差評価を得るために、有限要素法の基本的な誤差評価式 (3.6) と Poisson 方程式の解自体の性質を組み合わせた巧妙な Trick である。

また、(3.6) と (3.7) を比較すると、h に関するオーダーは H_0^1 誤差評価が 1 次であるのに対し、L^2 誤差評価は 2 次であることが分かる。

3.2 C_0, C_1 の具体的な値

C_1 の値については次のことが分かっている (Ladyzenskaya [6], 中尾 [8])。

Lemma 1.

Ω が区分的に滑らかな有界凸領域か、または、多角形領域であれば、(3.1) の解 $\xi \in H^2$ に対し、(3.3) の定数 C_1 は $C_1 = 1$ と取れる。すなわち、

$$|\xi|_{H^2} \leq \|g\|$$

が成り立つ。

一方、C_0 の値は、近似空間 S_h のとり方による。ここでは、簡単のため、$\Omega = (0,1) \times (0,1)$ とし、Ω を正方形要素に等分割する。なお、正方形要素による分割を T^h と書き、各正方形要素を $e, e_i \in T^h$ などと書くことにする。

また、x 軸方向の分割数を $n+1$ として、分割のパラメータ h を $h = \frac{1}{n+1}$ とおく。

次のような S_h に対しての C_0 の値について述べる。k を正整数として、

$$S_h \equiv \left\{ \phi \in C(\overline{\Omega}) \, ; \phi \mid_e \in Q^k(e), \ \forall e \in T^h \text{かつ } \phi = 0 \text{ on } \partial\Omega \right\} \tag{3.8}$$

を考える。

ここで、$\phi \mid_e$ とは、ϕ を e に制限したものを意味し、$Q^k(e)$ とは次のようなものである。

$$Q^k(e) \equiv \left\{ v = \sum_{i,j=0}^{k} \alpha_{ij} x^i y^j \ ; \ \alpha_{ij} \in \mathbf{R} \right\} \tag{3.9}$$

各 $\phi \in Q^k(e)$ は、x と y のそれぞれについてだけ見れば、k 次多項式であり、このような ϕ は双 k 次多項式と呼ばれている。

この S_h に対し、$k = 1$ のときは、$C_0 = 1/\pi$、$k = 2$ のときは、$C_0 = 1/2\pi$ に取れることが分かっている (中尾 [8], Schultz [14])。

したがって、次の補題が成り立つ。

Lemma 2.

$\Omega = (0,1) \times (0,1)$ で、(3.8) で定義される S_h に対して以下の基本的な誤差評価式が成り立つ。

$$\|\nabla (\xi - P_h \xi)\| \leq \begin{cases} \dfrac{1}{\pi} h \|g\| & (k=1) \\ \dfrac{1}{2\pi} h \|g\| & (k=2) \end{cases} \tag{3.10}$$

$$\|\xi - P_h \xi\| \leq \begin{cases} \dfrac{1}{\pi} h \|\nabla (\xi - P_h \xi)\| & (k=1) \\ \dfrac{1}{2\pi} h \|\nabla (\xi - P_h \xi)\| & (k=2) \end{cases} \tag{3.11}$$

これらの評価を元にして、楕円型偏微分方程式の解の精度保証を行う。

4　線形楕円型偏微分方程式に対する精度保証付き数値計算

4.1　考える問題とその有限要素解

次の問題を考える。

$$\begin{cases} -\Delta u = f(x, u, \nabla u) & \text{in } \Omega \subset \mathbf{R}^2 \\ u = 0 & \text{on } \partial\Omega \end{cases} \tag{4.1}$$

ここで、Ω は $\mathbf{R}^2$ 上の有界凸閉集合でその境界 $\partial\Omega$ は区分的に滑らかであるとする。
また、$u, \nabla u$ に関しては 1 次式であるような写像 f は次の仮定を満たすものとする。

A.1 $f : H_0^1 \ni u \mapsto f(\cdot, u, \nabla u) \in L^2$ は連続.

A.2 各有界集合 $U \subset H_0^1$ に対し、$f(\cdot, U, \nabla U) \equiv \{f(\cdot, u, \nabla u) \ ; \ u \in U\} \subset L^2$ は有界.

有限要素空間 S_h は区分的双 1 次多項式関数 ϕ_i で張られる空間で、その次元は N であるとする。
そして、H_0^1-projecton $P_h : H_0^1 \to S_h$ を、

$$(\nabla(\psi - P_h\psi), \nabla\phi)_{L^2} = 0, \qquad \forall\phi \in S_h$$

と定義する。また、$S_h^\perp$ を H_0^1 における S_h の直交補空間、すなわち

$$S_h^\perp \equiv \left\{ v^\perp \in H_0^1 \ ; \ \left(\nabla v^\perp, \nabla\phi\right) = 0, \quad \forall\phi \in S_h \right\}$$

とする。
(4.1) の弱解とは、次式を満たす $u \in H_0^1$ として定義できる。

$$(\nabla u, \nabla\psi)_{L^2} = (f(\cdot, u, \nabla u), \psi)_{L^2}, \quad \forall\psi \in H_0^1$$

また、(4.1) の有限要素近似解とは、次式を満たす $\hat{u}_h \in S_h$ のことであると定義される。

$$(\nabla\hat{u}_h, \nabla\phi)_{L^2} = (f(\cdot, u_h, \nabla u_h), \phi)_{L^2}, \quad \forall\phi \in S_h \tag{4.2}$$

(4.2) で表されるの有限要素法近似解 $\hat{u}_h \in S_h$ の係数ベクトルを $\vec{u} = (u_1, u_2, \cdots, u_N)^T \in \mathbf{R}^N$ とすると、$\vec{u}$ に関する連立一次方程式

$$D\vec{u} = \vec{a}$$

を得る。ここで、$D = (D_{ij})$ は $N \times N$ 行列、 $\vec{a} = (a_i)$ は N 次元ベクトルで、

$$\begin{aligned} D_{ij} &= (\nabla\phi_i, \nabla\phi_j)_{L^2} \\ a_i &= (f(\cdot, \hat{u}_h, \nabla\hat{u}_h), \phi_i)_{L^2} \end{aligned} \tag{4.3}$$

である。

4.2　不動点形式

この論文では不動点定理を利用して、解の精度保証を行う。そこで、(4.1) を不動点方程式 $u = F(u)$ の形に置き換えることにする。
Poisson 方程式 (3.1) を解くことを、g から ξ への対応関係とみなして、

$$\xi = K(g) \tag{4.4}$$

と書く (つまり、K は同次 Dicichlet 境界条件つき Laplacian の逆作用素である)。

これを用いて、F を

$$F(u) \equiv K(f(\cdot, u, \nabla u))$$

で定めれば、$u = F(u)$ が、問題 (4.1) を表すことになり、さらに $F : H_0^1 \to H_0^1$ はコンパクト作用素となる。なんとなれば、$K : L^2 \to H_0^1 \cap H^2$ は連続作用素で ($\because$ (3.3) より $|Kg|_{H^2} \leq \|g\|$)、埋め込み作用素 $\iota : H^2 \hookrightarrow H^1$ のコンパクト性 (Adams [1]) により、K は L^2 から H_0^1 へのコンパクト作用素である。したがって、$f : H_0^1 \to L^2$ が連続であると言う仮定から、F はコンパクト作用素であることが分かる。

こうして得られた不動点方程式に対して、次の Schauder の不動点定理 (増田 [7]) を適用する。

Theorem 1. (Schauder の不動点定理)

M を Banach 空間 X の空でない有界凸閉集合として、$T : M \to M$ をコンパクト作用素とする。このとき、T は不動点を持つ。

この定理によれば、有界凸閉集合であるような関数の集合 $U \subset H_0^1$ に対して、

$$F(U) \equiv \{v \ ; \ v = F(u), u \in U\}$$

と定義したとき、

$$F(U) \subset U$$

が成立すれば、$F(U)$ の中に不動点方程式の真の解が存在する。

4.3 候補者集合

真の解を含む可能性を持っている関数の集合を候補者集合と呼ぶことにする。問題 (4.1) の有限要素近似解 $\hat{u}_h$ の係数ベクトル $\vec{u}$ の第 i 成分 u_i を中心とした区間を考え、それを $u_i + W_i$ と書くことにする ($i = 1, 2, \cdots, N$)。つまり、W_i は $W_i = [-\overline{W_i}, \overline{W_i}]$, $(\overline{W_i} \geq 0)$ の形の区間である (この W_i のとり方はここでは決めていないことに注意)。

この区間 $u_i + W_i$ の中に係数が含まれるような S_h の元の集合を U_h とする。つまり、

$$\begin{aligned} U_h &\equiv \left\{ v_h \in S_h \ ; \ v_h = \sum_{i=1}^{N} v_i \phi_i, \ v_i - u_i \in W_i, \ (i = 1, 2, \cdots, N) \right\} \\ &= \left\{ \hat{u}_h + w_h \ ; \ w_h = \sum_{i=1}^{N} w_i \phi_i, \ w_i \in W_i, \ (i = 1, 2, \cdots, N) \right\} \end{aligned}$$

とおく。また、関数の微分を L^2 ノルムで計った半径 $\alpha \in \mathbf{R}^+$ の球を $U^\perp$ とする。つまり、

$$U^\perp \equiv \{v^\perp \in S_h^\perp \ ; \ \|\nabla v^\perp\| \leq \alpha\}$$

とおく。この定数 α も後で決定する。

これらより、候補者集合 $U \subset H_0^1$ を

$$\begin{aligned} U &\equiv U_h + U^\perp \\ &= \left\{ v \in H_0^1 \ ; \ v = v_h + v^\perp, \ v_h \in U_h, \ v^\perp \in U^\perp \right\} \end{aligned}$$

と定める。

4.4 検証条件

U に対して $F(U)$ を数値的に厳密に決定できればよいのだが、途中で解かねばならない Poisson 方程式 (3.1) が計算機上では近似的にしか解けない。したがって、$F(U)$ の代わりにこれを包含する集合 $\tilde{F}(U)$ を以下のように定め、代用する。

方程式 (3.1) を近似的に解くことを

$$P_h\xi = P_hK(g)$$

と書く。これを用いて U に対して、

$$P_hF(v) = P_hK(f(\cdot, v, \nabla v)), \quad v \in U \tag{4.5}$$

として、

$$F_h(U) \equiv \{f_h \in S_h \ ; \ f_h = P_hF(v), v \in U\}$$

および、

$$F^{\perp}(U) \equiv \left\{f^{\perp} \in S_h^{\perp} \ ; \ \|\nabla(f^{\perp})\| \leq \beta\right\} \tag{4.6}$$

と定める (定数 β についてはすぐ後で説明する)。$F_h(U)$ は $F(U)$ の S_h への丸めの集合、$F^{\perp}(U)$ は丸め誤差の集合と呼ばれる。

これを用いて、

$$\begin{aligned} \tilde{F}(U) &\equiv F_h(U) + F^{\perp}(U) \\ &= \left\{\tilde{f} \in H_0^1 \ ; \ \tilde{f} = f_h + f^{\perp}, \ f_h \in F_h(U), \ f^{\perp} \in F^{\perp}(U)\right\} \end{aligned}$$

とおく。すなわち、集合 $\tilde{F}(U)$ として、候補者集合と同じように、S_h の部分集合 $F_h(U)$ の各点に球がくっついた形をとることにする。

(4.6) の定数 β は

$$F(U) \subset \tilde{F}(U)$$

が成り立つように決める。具体的には、まず Poisson 方程式 (3.1) に対する誤差評価式 (3.10) から

$$\|\nabla\left(F(v) - P_hF(v)\right)\| \leq C_0h\|f(\cdot, v, \nabla v)\|, \quad \forall v \in U$$

が成り立つ。そこで、

$$\beta \equiv \sup_{v \in U} C_0h\|f(\cdot, v, \nabla v)\|$$

とおくと、$F(v) - P_hF(v) \in F^{\perp}(U)$ であるから、

$$\begin{aligned} F(v) &= P_hF(v) + (F(v) - P_hF(v)) \\ &\in P_hF(U) + F^{\perp}(U) = \tilde{F}(U) \end{aligned}$$

となる。故に $F(U) \subset \tilde{F}(U)$ が成り立つ。

こうして定義した $\tilde{F}(U)$ について、

$$\tilde{F}(U) \subset U \tag{4.7}$$

が成り立てば、不動点定理が成り立つのだが、U と $\tilde{F}(U)$ が同様の構造をもつことから、(4.7) が成り立つための十分条件として、

$$\begin{cases} F_h(U) \subset U_h \\ F^{\perp}(U) \subset U^{\perp} \end{cases} \tag{4.8}$$

を得る。これを問題 (4.1) の解の検証条件と呼ぶことにする。

4.5 区間演算

上の検証条件のうち、最初のものをチェックするためには、$F_h(U)$ を評価しなければならない。U が無限集合であることから、一見して無限個の近似解の計算が必要に思われるかもしれないが、これは区間演算を利用することで解決できる。

(4.5) により $f_h = P_h F(v)$ は以下を満たす S_h の元である。

$$(\nabla f_h, \nabla\phi_i)_{L^2} = (f(\cdot, v, \nabla v), \phi_i)_{L^2}, \quad v \in U \tag{4.9}$$

したがって、

$$(\nabla(f_h - \hat{u}_h), \nabla\phi_h)_{L^2} = (f(\cdot, v, \nabla v) - f(\cdot, \hat{u}_h, \nabla\hat{u}_h), \phi_h)_{L^2}, \quad v \in U \tag{4.10}$$

であり、そして、(4.10) の左辺に含まれる $v \in U$ を

$$v = v_h + v^\perp \ \left(v_h \in U_h,\ v^\perp \in U^\perp\right) \tag{4.11}$$

と分解し、さらに U_h の定義に基づき v_h を $v_h = \hat{u}_h + w_h$ と書く。これより、f_h の係数ベクトルを $\vec{f}$ とすると

$$D(\vec{f} - \vec{u}) = \vec{b}$$

を得る。ここで、D は (4.3) で表されるものであり、$\vec{b} = (b_i)$ は

$$b_i = (f(\cdot, v_h + v^\perp, \nabla(v_h + v^\perp)) - f(\cdot, \hat{u}_h, \nabla\hat{u}_h), \phi_i)_{L^2}$$

である。

いま、任意の行列 $M \in \mathbf{R}^{n\times n}$ に対して、M の各要素の絶対値を要素としてもつ行列を $|M|$ と書くことにする。また、各成分が 0 中心の対象な区間値であるような区間ベクトル $\vec{Y} \in IR^n$ をとり、これに対し連立 1 次方程式

$$M\vec{x} = \vec{y}, \quad \left(\vec{y} \in \vec{Y}\right)$$

を考える。このとき、M が正則であれば、

$$\vec{x} \in |M^{-1}|\vec{Y} \tag{4.12}$$

が成り立つ事が知られている (Alefeld [2])。

したがって、$\vec{b}$ を内包するような、各成分が 0 中心の対称な区間値であるような区間ベクトル $\vec{C}$ に対して (i.e. $\vec{C} = (C_i),\ C_i = [-c_i, c_i]$ で、$b_i \in [-c_i, c_i]$ が成り立つような $\vec{C}$ に対して)、$D(\vec{f} - \vec{u}) = \vec{b}$ の解 $\vec{f} - \vec{u}$ は

$$\vec{f} - \vec{u} \in |D^{-1}|\vec{C}$$

を満たすことが分かる。

なお、$\vec{C}$ の性質より、区間ベクトル $\vec{Y} \equiv |D^{-1}|\vec{C}$ も各成分は 0 中心の対称な区間値であり、$\vec{Y} = (Y_i),\ Y_i = [-\overline{Y}_i, \overline{Y}_i]$ と書くことにする。

ゆえに、集合 $F_h(U)$ は、

$$F_h(U) \subset \left\{ f_h \in S_h\ ;\ \vec{f} - \vec{u} \in \vec{Y} \right\}$$

と包み込める。一方、集合 $F^\perp(U)$ については、

$$F^\perp(U) = \left\{ f^\perp \in S_h^\perp\ ;\ \|\nabla f^\perp\| \le \sup_{v\in U} C_0 h \|f(\cdot, v, \nabla v)\| \right\}$$

したがって、次の条件が検証条件 (4.8) の十分条件となる。

$$\begin{cases} \overline{Y}_i \le \overline{W}_i \quad (i = 1, 2, \cdots, N) \\ \sup\limits_{v\in U} C_0 h \|f(\cdot, v, \nabla v)\| \le \alpha \end{cases} \tag{4.13}$$

4.6 具体的な数値例

ここでは次のような線形楕円型偏微分方程式を元に、精度保証付き数値計算の具体的な手順を述べる。

$$\begin{cases} -\Delta u = \pi u + (2\pi - 1)\sin \pi x \sin \pi y, & \text{in } \Omega = (0,1) \times (0,1) \\ u = 0, & \text{on } \partial\Omega \end{cases} \tag{4.14}$$

この問題の解は既知であり、$u(x,y) = \dfrac{1}{\pi}\sin \pi x \sin \pi y$ である。

4.6.1 有限要素法

$(0,1)$ 区間の x 方向を $n+1$ 等分し、分割点に以下のように番号をつける。

$$0 = x_0 < x_1 < x_2 < \cdots < x_n < x_{n+1} = 1$$

y 方向も $n+1$ 等分し、同様に分割点に番号をつける。

$h = \frac{1}{n+1}$ とする。(i,j) 番目の節点 (x_i, y_j) に対し、その点で 1 をとり、他の節点で 0 となる区分的双一次関数 ϕ_{ij} を考える $(i,j = 1, \cdots, n)$。つまり、

$$\phi_{ij}(x_k, y_l) = \begin{cases} 1 & (i = k \text{ かつ } j = l) \\ 0 & (\text{otherwise}) \end{cases} \qquad (i,j = 1, \cdots, n)$$

を満たすものである。

具体的には以下の式で与えられる。

$$\phi_{ij}(x,y) = \begin{cases} \dfrac{1}{h^2}(x - x_{i-1})(y - y_{j-1}) & (x_{i-1} \le x \le x_i,\ y_{j-1} \le y \le y_j) \\ \dfrac{1}{h^2}(x - x_{i-1})(y_{j+1} - y) & (x_{i-1} \le x \le x_i,\ y_j \le y \le y_{j+1}) \\ \dfrac{1}{h^2}(x_{i+1} - x)(y - y_{j-1}) & (x_i \le x \le x_{i+1},\ y_{j-1} \le y \le y_j) \\ \dfrac{1}{h^2}(x_{i+1} - x)(y_{j+1} - y) & (x_i \le x \le x_{j+1},\ y_j \le y \le y_{j+1}) \\ 0 & (\text{otherwise}) \end{cases}$$

有限要素空間 S_h は ϕ_{ij} $(i,j = 1,2,\cdots,n)$ で張られるものを考える。S_h の次元は n^2 である。

代表要素 $e : (x_i, x_{i+1}) \times (y_j, y_{j+1})$ 上の局所的な節点の番号付けを図 1 のようにする。

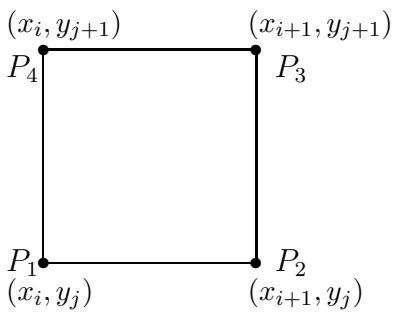


図 1: 代表要素 e の局所的番号 1

代表要素 e 上の基底関数 φ_i $(i=1,2,3,4)$ は以下を満たすようにとる。

$$\varphi_i(P_j)=\delta_{ij}\quad (i,j=1,2,3,4)$$

具体的には

$$\varphi_1=\frac{1}{h^2}(x_{i+1}-x)(y_{j+1}-y)$$

$$\varphi_2=\frac{1}{h^2}(x-x_i)(y_{j+1}-y)$$

$$\varphi_3=\frac{1}{h^2}(x-x_i)(y-y_j)$$

$$\varphi_4=\frac{1}{h^2}(x_{i+1}-x)(y-y_j)$$

となる。

問題 (4.14) の弱解 $u\in H_0^1$ は

$$(\nabla u,\nabla\psi)_{L^2}-\pi\,(u,\psi)_{L^2}=((2\pi-1)\sin\pi x\sin\pi y,\psi)_{L^2}\,,\quad \forall\psi\in H_0^1$$

であるから、有限要素近似解として $\hat{u}_h\in S_h$ s.t.

$$(\nabla\hat{u}_h,\nabla\phi_{ij})_{L^2}-\pi\,(\hat{u}_h,\phi_{ij})_{L^2}=((2\pi-1)\sin\pi x\sin\pi y,\phi_{ij})_{L^2}\,,\quad (i,j=1,\cdots,n)$$

を採用し、$\hat{u}_h=\sum_{i,j=1}^{n}u_{ij}\phi_{ij},\ u_{ij}\in\mathbf{R}\ \ (i,j=1,\cdots,n)$ を代入すると $\vec{u}=(u_{11},\cdots,u_{nn})^T\in\mathbf{R}^{n^2}$ に関する連立 1 次方程式

$$D\vec{u}-\pi L\vec{u}=\vec{a}$$

を得る。ここに、$D=(D_{ijkl}),L=(L_{ijkl})$ は $n^2\times n^2$ 行列、$\vec{a}=(a_{ij})$ は n^2 次元ベクトルで、その成分は

$$\begin{aligned}
&D_{ijkl}=(\nabla\phi_{ij},\nabla\phi_{kl})_{L^2}\\
&L_{ijkl}=(\phi_{ij},\phi_{kl})_{L^2}\qquad\qquad (i,j,k,l=1,2,\cdots,n)\\
&a_{ij}=((2\pi-1)\sin\pi x\sin\pi y,\phi_{ij})_{L^2}
\end{aligned}$$

である。

代表要素 e 上において $(D_e)_{ij}\equiv(\nabla\varphi_i,\nabla\varphi_j)_{L^2(e)}$、$(L_e)_{ij}\equiv(\varphi_i,\varphi_j)_{L^2(e)}$、$(a_e)_i\equiv(\sin\pi x\sin\pi y,\varphi_i)_{L^2(e)}$ はそれぞれ

$$D_e=\frac{1}{6}\begin{pmatrix}4&-1&-2&-1\\&4&-1&-2\\&\mathrm{Sym}&4&-1\\&&&4\end{pmatrix}$$

$$L_e=\frac{h^2}{36}\begin{pmatrix}4&2&1&2\\&4&2&1\\&\mathrm{Sym}&4&2\\&&&4\end{pmatrix}$$

$$\vec{a}_e = \begin{pmatrix} \frac{1}{\pi^2 h^2}\left\{-\frac{1}{\pi}\left(\sin\pi x_{i+1} - \sin\pi x_i\right) + h\cos\pi x_i\right\}\left\{-\frac{1}{\pi}\left(\sin\pi y_{i+1} - \sin\pi y_i\right) + h\cos\pi y_i\right\} \\ \frac{1}{\pi^2 h^2}\left\{\frac{1}{\pi}\left(\sin\pi x_{i+1} - \sin\pi x_i\right) - h\cos\pi x_{i+1}\right\}\left\{-\frac{1}{\pi}\left(\sin\pi y_{i+1} - \sin\pi y_i\right) + h\cos\pi y_i\right\} \\ \frac{1}{\pi^2 h^2}\left\{\frac{1}{\pi}\left(\sin\pi x_{i+1} - \sin\pi x_i\right) - h\cos\pi x_{i+1}\right\}\left\{\frac{1}{\pi}\left(\sin\pi y_{i+1} - \sin\pi y_i\right) - h\cos\pi y_{j+1}\right\} \\ \frac{1}{\pi^2 h^2}\left\{-\frac{1}{\pi}\left(\sin\pi x_{i+1} - \sin\pi x_i\right) + h\cos\pi x_i\right\}\left\{\frac{1}{\pi}\left(\sin\pi y_{i+1} - \sin\pi y_i\right) - h\cos\pi y_{j+1}\right\} \end{pmatrix}$$

となる。

ここで、

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ & a_{22} & a_{23} & a_{24} \\ & \mathrm{Sym} & a_{33} & a_{34} \\ & & & a_{44} \end{pmatrix}$$

は対称行列 $A = A^T$ を表す。

4.6.2 区間演算

(4.9) ~ (4.11) より

$$\begin{aligned} (\nabla(f_h - \hat{u}_h), \nabla\phi_{ij})_{L^2} &= (\pi(v - \hat{u}_h) + (2\pi - 1)\sin\pi x \sin\pi y, \phi_{ij})_{L^2} \\ &= (\pi(\hat{u}_h + w_h + v^\perp - \hat{u}_h) + (2\pi - 1)\sin\pi x \sin\pi y, \phi_{ij})_{L^2} \\ &= (\pi(w_h + v^\perp), \phi_{ij})_{L^2}, \quad (i, j = 1, \cdots, n) \end{aligned}$$

を得る。

$b_{ij} = (\pi(w_h + v^\perp), \phi_{ij})_{L^2}$ $(i, j = 1, \cdots, n)$ の評価は以下のとおりである。

$w_h = \sum_{i,j=1}^{n} w_{ij}\phi_{ij},\ w_{ij} \in W_{ij} = [-\overline{W}_{ij}, \overline{W}_{ij}]$ $(i, j = 1, \cdots, n)$ であるから、

$$|(w_h, \phi_{ij})_{L^2}| \le \sum_{k,l=1}^{n} \left|(\phi_{ij}, \phi_{kl})_{L^2}\right| \overline{W}_{kl}$$

が成り立つ。一方、Schwarz の不等式より、

$$|(v^\perp, \phi_{ij})_{L^2}| \le \|v^\perp\|\,\|\phi_{ij}\|$$

が成り立ち、さらに Poisson 方程式 (3.1) に対する誤差評価式 (3.11) より

$$\|v^\perp\| \le \frac{1}{\pi} h \|\nabla v^\perp\| \tag{4.15}$$

が成り立つので、

$$|(v^\perp, \phi_{ij})_{L^2}| \le \frac{1}{\pi} h \|\nabla v^\perp\|\,\|\phi_{ij}\| \le \frac{1}{\pi} h\alpha \|\phi_{ij}\|$$

を得る。したがって、

$$|b_{ij}| \le \tilde{b}_{ij} = \pi \sum_{k,l=1}^{n} |L_{ijkl}|\overline{W}_{kl} + h\alpha\|\phi_{ij}\|$$

と b_{ij} は評価できる。

これより、

$$C_{ij} = [-\tilde{b}_{ij}, \tilde{b}_{ij}]$$

によって区間ベクトル $\vec{C}$ を定義すると、$D(\vec{f} - \vec{u}) = \vec{b}$ の解 $\vec{f} - \vec{u}$ は

$$\vec{f} - \vec{u} \in |D^{-1}|\vec{C}$$

を満たすことになる。

次に、

$$\sup_{v \in U} \|f(v)\| = \sup_{v \in U} \|\pi v + (2\pi - 1) \sin \pi x \sin \pi y\|$$

の評価だが、これは次のように考える。

(4.11) と同様に、$v = v_h + v^{\perp}$ と分解して、さらに (4.15) より、

$$\sup_{v \in U} \|f(v)\| \leq \pi \sup_{v_h \in U_h} \|v_h\| + h\alpha + (2\pi - 1)\| \sin \pi x \sin \pi y\|$$

とする。$\sup_{v_h \in U_h} \|v_h\|$ の計算機上での評価は、

$$\begin{aligned} \sup_{v_h \in U_h} \|v_h\| &= \sup \|U_h\| = \sup \sqrt{(U_h, U_h)_{L^2}} \\ &= \sup \sqrt{\sum_{i,j,k,l=1}^{n} (u_{ij} + W_{ij})\, L_{ijkl}\, (u_{kl} + W_{kl})} \end{aligned}$$

と計算する。

4.6.3 具体的なアルゴリズム

Step.1 正数 $\delta \ll 1$ を定める。後の数値実験では $\delta = 1.0e^{-9}$ とした。

Step.2 はじめに $W_{ij}^{(0)} = [0, 0] \quad (i, j = 1, \cdots, n)$, および、$\alpha^{(0)} = 0$ ととる。これらによって定まる候補者集合を $U^{(0)}$ とする。すなわち、$U^{(0)} \equiv \hat{u}_h$ とおく。

Step.3 $W_{ij}^{(k)}$, $\alpha^{(k)}$ および、これらによって定まる候補者集合 $U^{(k)}$ に対して、条件 (4.13) をチェックする。満足すれば反復を終了する。このとき求める解は $U^{(k)}$ の中に存在する。そうでなかったら、次のステップへ。

Step.4 $W_{ij}^{(k)}$, $\alpha^{(k)}$, $U^{(k)}$ と、これらから定まる $\vec{C}^{(k)}$ を用いて、

$$W_{ij}^{(k+1)} = (1 + \delta) \left(|D^{-1}| \vec{C}^{(k)} \right)_{ij} \quad (i, j = 1, \cdots, n),$$

$$\alpha^{(k+1)} = (1 + \delta) \frac{1}{\pi} h \left(\pi \sup_{v_h \in U_h^{(k)}} \|v_h\| + \pi \alpha^{(k)} + (2\pi - 1)\| \sin \pi x \sin \pi y\| \right)$$

によって、$W_{ij}^{(k+1)}$, $\alpha^{(k+1)}$ を定め、これから新しい候補者集合 $U^{(k+1)}$ を定義して、前のステップに戻る。

ここで、正数 δ を用いたのは、$U^{(k)} \subset U^{(k+1)}$ と言う拡大のスピードの鈍化を抑えて、検証条件の成り立つ集合にいたるまで膨らむのを助けようというのである。このことを δ-inflation という。

あらかじめ決めておいた反復回数の最大に達するまでに検証条件を満足する候補者集合が得られなければ、検証は失敗である。その場合には h をより小さくとって $C_0 h$ の値を小さくし、検証条件が成り立ちやすいようにしてやり直すか、あるいは次節 5.4 節で述べる擬似 Newton 法を用いることになる。

4.6.4 数値計算結果

精度保証付き数値計算においては、数値演算は機械区間演算を用いることになる。そこで、この数値実験では「**PROFIL**」という数値計算ソフトウェアを用いた。この **PROFIL** は Humburg 大学で開発された C++ 言語による汎用精度保証ソフトウェアで、フリーソフトとして公開されている。

この **PROFIL** では、INT(整数) 型、REAL(実数) 型、INTERVAL(区間) 型とこれらの型のベクトルおよび行列、COMPLEX(複素数) 型の演算をサポートしている。さらに、Rump の方法により連立一次 (区間) 方程式の解をもとめるルーチンも含んでいる。この **PROFIL** に関するドキュメントはインターネット上[1] で入手できる。

先に述べたように、問題 (4.14) の解は既知である。したがって、数値計算で求まった解の区間の中に、厳密解が含まれているかどうかは容易に判定できる。

この問題の解の概形は図 2 のようなものである。

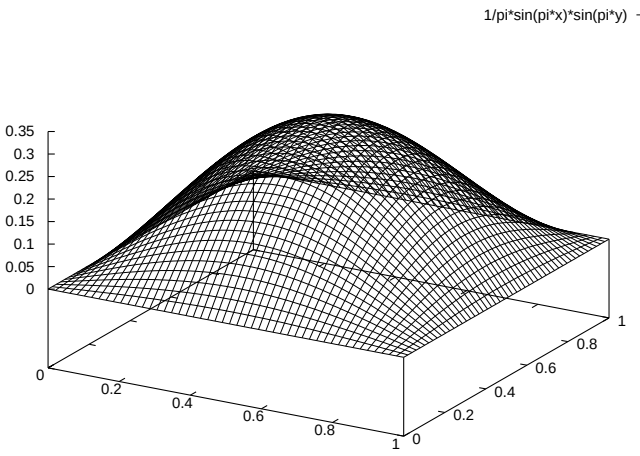


図 2: 問題 (4.14) の解の概形

この厳密解の計算では浮動小数点数演算は倍精度を用いている。IEEE 倍精度浮動小数点数は 10 進法で 16 桁程度の精度がある。

この厳密解の計算における丸め誤差は無視している。しかし、倍精度演算における丸め誤差のオーダーは高々 10^{-10} 程度なので、厳密解が含まれているかどうかの判定には、影響しないであろう。

まず、分割数が 5 のときの結果を示す。このとき、分割のパラメータ $h = 0.2$ となり、総節点数は 36 である。

反復回数 15 回で反復を終了し、表 1 (p.21) のような結果を得た。

ここで、$U(i)$ とは、候補者集合 U の有限次元部分

$$U_h = \left\{ \hat{u}_h + w_h \ ; \ w_h = \sum_{i=1}^{n^2} w_i \phi_i, \ w_i \in W_i, \ (i = 1, 2, \cdots, n^2) \right\}$$

の係数ベクトル $u_i + W_i$ である。解の値が 0 になっている節点は、Dirichlet 条件に対応する境界上の節点であることを注意しておく。

[1] http://www.ti3.tu-harburg.de/Software/PROFILEnglisch.html

節点	$U(i)$	厳密解	厳密解を含むか	$U(i)$ の幅
1	[-0,0]	0	◎	0
2	[-0,0]	0	◎	0
3	[-0,0]	0	◎	0
4	[-0,0]	0	◎	0
5	[-0,0]	0	◎	0
6	[-0,0]	0	◎	0
7	[-0,0]	0	◎	0
8	[0.107299,0.118596]	0.109973	◎	0.011297
9	[0.174860,0.190646]	0.177941	◎	0.015786
10	[0.174860,0.190646]	0.177941	◎	0.015786
11	[0.107299,0.118596]	0.109973	◎	0.011297
12	[-0,0]	0	◎	0
13	[-0,0]	0	◎	0
14	[0.174860,0.190646]	0.177941	◎	0.015786
15	[0.284237,0.307164]	0.287914	◎	0.022927
16	[0.284237,0.307164]	0.287914	◎	0.022927
17	[0.174860,0.190646]	0.177941	◎	0.015786
18	[-0,0]	0	◎	0
19	[-0,0]	0	◎	0
20	[0.174860,0.190646]	0.177941	◎	0.015786
21	[0.284237,0.307164]	0.287914	◎	0.022927
22	[0.284237,0.307164]	0.287914	◎	0.022927
23	[0.174860,0.190646]	0.177941	◎	0.015786
24	[-0,0]	0	◎	0
25	[-0,0]	0	◎	0
26	[0.107299,0.118596]	0.109973	◎	0.011297
27	[0.174860,0.190646]	0.177941	◎	0.015786
28	[0.174860,0.190646]	0.177941	◎	0.015786
29	[0.107299,0.118596]	0.109973	◎	0.011297
30	[-0,0]	0	◎	0
31	[-0,0]	0	◎	0
32	[-0,0]	0	◎	0
33	[-0,0]	0	◎	0
34	[-0,0]	0	◎	0
35	[-0,0]	0	◎	0
36	[-0,0]	0	◎	0

表 1: 分割数 5 のときの結果

また、このとき区間 W_i の幅の最大 $= 0.0229270187$、無限次元部分の半径 ; $\alpha = 0.2026268413$ という結果を得た。プログラム中では α も区間で求めているが、見やすさの為、これより α は区間の上限を表示することにする。

精度保証の理論によれば、有限次元部分 U_h の各係数に半径 α の無限次元部分の球がついたような閉集合の中に真の解が存在するのであった。

したがって、全ての節点において真の解を含んでいることが確認できた。

また、分割数が 20 のときの $y = 0.5$ における区間 $U(i)$ の上限と下限のグラフは図 3 のようになる。このグラフにおいても、区間 $U(i)$ の中に厳密解が含まれていることを確認できる。

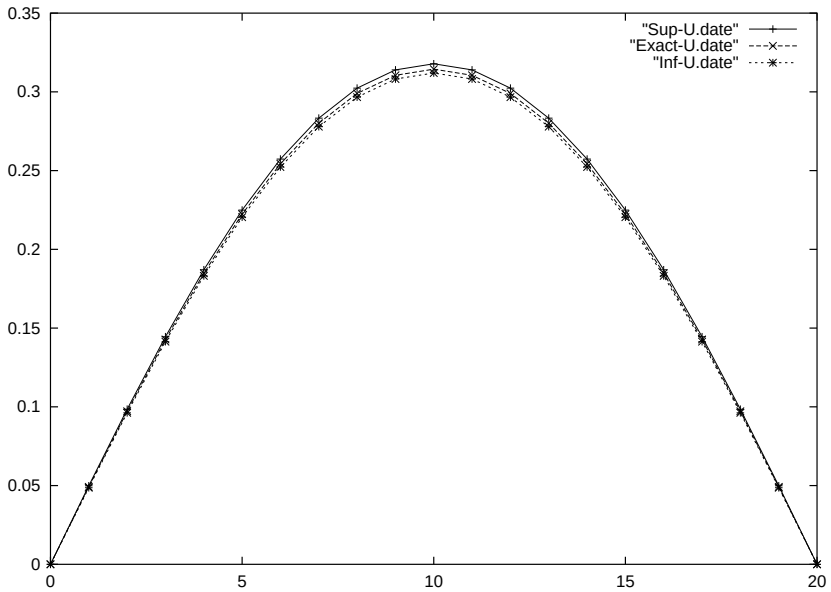


図 3: $y = 0.5$ における U_h のグラフ 1

最後に、分割数を増やすにしたがって、区間 W_i の幅の最大と、無限次元部分の半径 α がどのように減少しているかを調べた (表 2)。

分割数	W_i の幅の最大	α
5	0.0229270187	0.2026268413
10	0.0119884908	0.1004889612
20	0.0059481067	0.0501017982
40	0.0029671370	0.0250229016

表 2: W_i と α

横軸を分割数、縦軸を W_i の幅の最大, α とした両対数グラフは図 4 のようになる。

W_i の幅の最大と α のグラフはともに直線状で、その傾きはそれぞれ、

$$\frac{\log(0.0029671) - \log(0.0119885)}{\log(40) - \log(10)} = -1.0072533 \approx -1.0$$

$$\frac{\log(0.0250229) - \log(0.1004890)}{\log(40) - \log(10)} = -1.0028880 \approx -1.0$$

となり、W の幅、α ともに、h のオーダーで減少しているのがわかり、したがって、分割数を上げれば、かなり精度のよい存在範囲が得られることが期待できる。

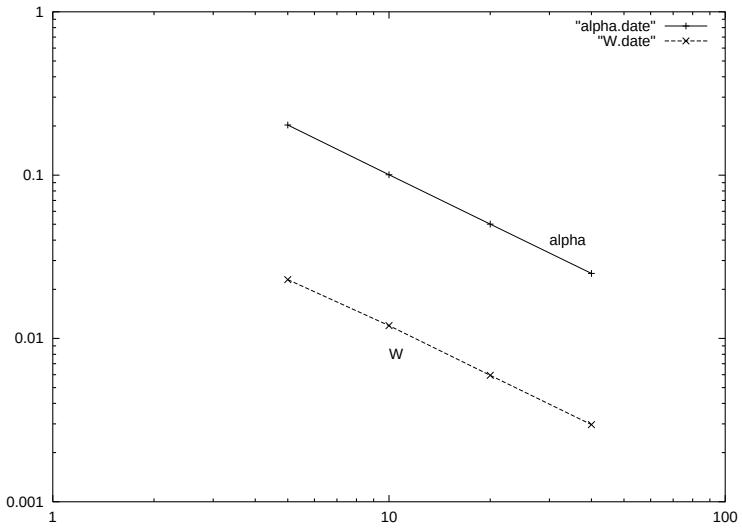


図 4: 分割数と W の幅,α の両対数グラフ

4.6.5 非強圧的線形楕円型偏微分方程式に対する数値実験

ここでは、次のような非強圧的線形楕円型偏微分方程式に対する精度保証付き数値計算を行う。

$$\begin{cases} -\Delta u = 20xyu + (2\pi - 1)\sin \pi x \sin \pi y, & \text{in } \Omega = (0,1) \times (0,1) \\ u = 0, & \text{on } \partial\Omega \end{cases} \tag{4.16}$$

解析的な理論はほぼ問題 (4.14) と同様なので、省略する。
この問題の解 $\hat{u}_h + W$ の概形は図 5 のとおりである。

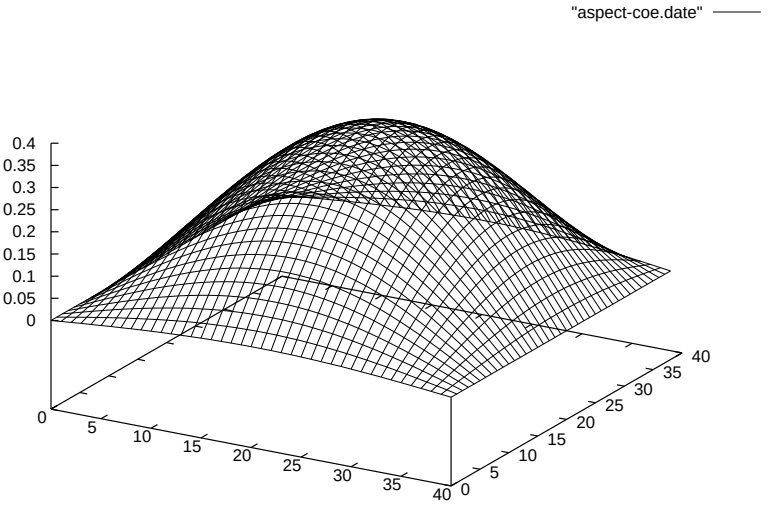


図 5: 問題 (4.16) の解の概形

分割数 $5 \sim 40$ について検証が成功した。表 3 にそのときの反復回数と W_i の幅の最大、α の値を示す。

分割数	反復回数	W の幅の最大	α
5	22	0.316136	0.365189
10	18	0.137819	0.158328
20	16	0.065701	0.075040
40	15	0.032077	0.036668

表 3: 非強圧的線形楕円型方程式に対する検証

また、分割数が 40 のときの $y = 0.5$ における区間 $U(i)$ の上限と下限のグラフは図 6 のようになる。

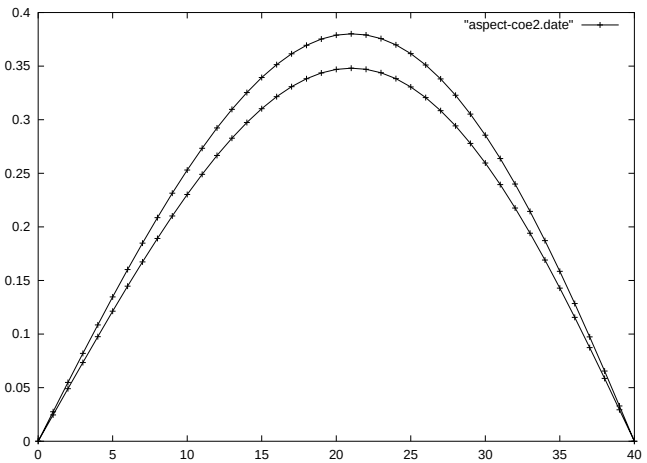


図 6: $y = 0.5$ における U_h のグラフ 2

厳密解は、この 2 つの曲線にさらに H_0^1 誤差 0.03667 を付加した領域に存在する。

次に、問題 (4.16) における u の項の係数 $20xy$ を $-60xy \sim 60xy$ と変えて実験してみる (分割数は 20 とする)。その結果を表 4(p.25) に示す。

係数の絶対値が 50 を超えると検証に失敗するのは、$\|f(u)\|$ の値が大きくなりすぎ、検証条件が成立しないためと思われる。

そこで、後に擬似 Newton 法を用いた解の検証を試みる。

係数	反復回数	W の幅の最大値	α の上限
-60	Verification Failure		
-55	Verification Failure		
-50	144	1.270428	0.213156
-45	67	0.509131	0.122546
-40	44	0.286977	0.095444
-35	32	0.182364	0.081816
-30	24	0.121353	0.073130
-25	20	0.081504	0.066697
-20	16	0.053736	0.061375
-15	13	0.033627	0.056572
-10	11	0.018873	0.051922
-5	8	0.007934	0.047157
0	2	2.680078e-13	0.042042
5	8	0.008060	0.047908
10	11	0.019991	0.054999
15	13	0.037918	0.063791
20	16	0.065701	0.075040
25	20	0.110004	0.090019
30	25	0.184293	0.111060
35	32	0.318594	0.142934
40	44	0.592889	0.197186
45	68	1.290934	0.310723
50	145	4.174006	0.700325
55	Verification Failure		
60	Verification Failure		

表 4: 係数の変化

5　半線形楕円型偏微分方程式に対する精度保証付き数値計算

前節で述べたように、問題 (4.1) に対する精度保証付き数値計算を行う際には、誤差評価式を用いた $\|\nabla\left(F(u)-P_hF(u)\right)\|$ の評価が重要な役割を果たす。具体的には

$$\|\nabla\left(F(u)-P_hF(u)\right)\| \leq C_0h\|f(u)\|$$

である。

だが、$f(u)$ が高次多項式の非線形項を持つような場合には、右辺が大きくなってしまい検証が成功しない場合もある。そのような場合についての精度保証付き数値計算の行い方を説明する。

考える問題の形式は 4 章と同様に次のようなものとする。

$$\begin{cases} -\Delta u = f(x,u,\nabla u) & \text{in } \Omega \subset \mathbf{R}^2 \\ u = 0 & \text{on } \partial\Omega \end{cases} \tag{5.1}$$

ここで、Ω は $\mathbf{R}^2$ 上の有界凸領域でその境界 $\partial\Omega$ は区分的に滑らかであるとする。

また、u に関しての高次多項式である非線形写像 f についての仮定も 4 節と同様に [A.1],[A.2] を満たすものとし、$f(u) \equiv f(\cdot,u,\nabla u)$ とおく。

5.1　残差誤差と残差方程式

問題 (5.1) の弱形式は次式で与えられる。

$$(\nabla u, \nabla\psi)_{L^2} = (f(u),\psi)_{L^2}, \quad \forall\psi \in H_0^1 \tag{5.2}$$

また、(5.2) の有限要素近似解は次式を満たす $u_h^* \in S_h$ のことであると定義される。

$$(\nabla u_h^*, \nabla\phi)_{L^2} = (f(u_h^*),\phi)_{L^2}, \quad \forall\phi \in S_h \tag{5.3}$$

何らかの方法で求めた (5.2) の近似解 $\hat{u}_h$ (通常は Newton 法で求めた (5.3) の近似解) を用いた残差反復を行う。

(5.2) より

$$(\nabla\left(u-\hat{u}_h\right),\nabla\psi)_{L^2} = (f(u)-f(\hat{u}_h),\psi)_{L^2} + (f(\hat{u}_h),\psi)_{L^2} - (\nabla\hat{u}_h,\nabla\psi)_{L^2}, \quad \forall\psi \in H_0^1$$

このとき、$v_0, w \in H_0^1$ s.t.

$$(\nabla v_0, \nabla\psi)_{L^2} = (f(\hat{u}_h),\psi)_{L^2} - (\nabla\hat{u}_h,\nabla\psi)_{L^2}, \quad \forall\psi \in H_0^1 \tag{5.4}$$

$$(\nabla w, \nabla\psi)_{L^2} = (f(\hat{u}_h+v_0+w)-f(\hat{u}_h),\psi)_{L^2}, \quad \forall\psi \in H_0^1 \tag{5.5}$$

を用いて、$u = \hat{u}_h + v_0 + w$ とすると、(5.2) を満たす。

(5.4) を満たす v_0 を残差誤差、式 (5.5) を残差方程式と言う。

(5.4) は同次 Dirichlet 条件付き Poisson 方程式なので (5.4) の解 $v_0 \in H_0^1$ は確かに存在する。それゆえ、右辺が十分小さいだろうと思われる (5.5) の解の存在を示せばよい。

この (5.5) の検証のため、(5.4) より $\|\nabla v_0\|$ を評価しなくてはならない。そして、(5.5) に対し、近似解を 0 に取った場合の精度保証付き数値計算を適用する。

5.2 $\|\nabla v_0\|$ の a priori 評価

v_h を v_0 の S_h への H_0^1-projection とする。つまり、

$$(\nabla(v_0 - v_h), \nabla\phi)_{L^2} = 0, \quad \forall\phi \in S_h$$

を満たすものとする。

このとき、次が成り立つ。

$$(\nabla v_h, \nabla\phi)_{L^2} = (f(\hat{u}_h), \phi)_{L^2} - (\nabla\hat{u}_h, \nabla\phi)_{L^2}, \quad \forall\phi \in S_h \tag{5.6}$$

$\|\nabla v_h\|$, $\|v_h\|$ は計算機上で評価でき、さらに $\hat{u}_h$ は有限要素近似解 u_h^* を十分よく近似しているはずなので、値はとても小さくなる。

一方、

$$\begin{cases} \Delta\eta = -f(\hat{u}_h) & \text{in } \Omega \\ \eta = 0 & \text{on } \partial\Omega \end{cases}$$

という Poisson 方程式の弱解 $\eta \in H_0^1 \cap H^2$ を用いると、

$$(\nabla v_0, \nabla\psi)_{L^2} = (\nabla\eta, \nabla\psi)_{L^2} - (\nabla\hat{u}_h, \nabla\psi)_{L^2}, \quad \forall\psi \in H_0^1$$

を得る。これにより、

$$(\nabla\eta - \nabla(\hat{u}_h + v_h), \nabla\phi)_{L^2} = 0, \quad \forall\phi \in S_h$$

が成り立つので、$\hat{u}_h + v_h$ は η の S_h への H_0^1-projection であることが分かる。このとき、誤差評価式 (3.10) より

$$\|\nabla(\eta - (\hat{u}_h + v_h))\| \le C_0 h \|f(\hat{u}_h)\|$$

が成り立つので、

$$\|\nabla v_0\| \le C_0 h \|f(\hat{u}_h)\| + \|\nabla v_h\| \tag{5.7}$$

となる。

次に、

$$\begin{cases} \Delta\xi = -v_0 & \text{in } \Omega \\ \xi = 0 & \text{on } \partial\Omega \end{cases}$$

を考え、ξ_h を ξ の S_h への H_0^1-projction とする。

このとき、

$$(\nabla v_0, \nabla\xi_h)_{L^2} = (\nabla v_h, \nabla\xi_h)_{L^2} = (\nabla v_h, \nabla\xi)_{L^2} = (v_h, v_0)_{L^2}$$

より

$$\begin{aligned} \|v_0\|^2 &= (v_0, v_0)_{L^2} = (\nabla v_0, \nabla\xi)_{L^2} = (\nabla v_0, \nabla(\xi - \xi_h))_{L^2} + (\nabla v_0, \nabla\xi_h)_{L^2} \\ &\le \|\nabla v_0\| \|\nabla(\xi - \xi_h)\| + \|v_0\| \|v_h\| \end{aligned}$$

が成り立つ。誤差評価式 (3.10) より、$\|\nabla(\xi - \xi_h)\| \le C_0 h \|v_0\|$ なので、結局

$$\|v_0\| \le C_0 h \|\nabla v_0\| + \|v_h\| \tag{5.8}$$

を得る。

5.3 $\|\nabla v_0\|$ の a posteriori 評価

前小節で $\|\nabla v_0\|$ の a priori 評価を得たわけであるが、通常は a priori 評価よりも a posteriori 評価のほうが小さいので、a posteriori 評価によって、残差誤差 $\|\nabla v_0\|$ の改善を図る。

そこで、まず平滑化 (smoothing) を考える。

$$S_h \equiv \left\{ \phi \in C(\overline{\Omega}) \, ; \phi \mid_e \in Q^k(e), \ \forall e \in T^h \text{ かつ } \phi = 0 \text{ on } \partial\Omega \right\}$$

に対して、

$$S_h^* \equiv \left\{ \phi \in C(\overline{\Omega}) \, ; \phi \mid_e \in Q^k(e), \ \forall e \in T^h \right\}$$

を考える。ここで、k は正整数で、$Q^k(e)$ は (3.9) で表される双 k 次多項式の集合である。

このような S_h^* を構成するには、S_h の基底に境界上の節点に対応する基底を追加したものを S_h^* とすれば良い。

この S_h^* は H_0^1 に属さず、$S_h^* \subset H^1$ となる。また、S_h^* の次元を M とする (S_h の次元は N)。

$P_L : L^2 \to S_h^*$ を L^2-projection として、2 次元ベクトル値関数 $\overline{\nabla}\hat{u}_h \in S_h^* \times S_h^*$ を次のように定義する。

$$\overline{\nabla}\hat{u}_h = \left(P_L \frac{\partial}{\partial x}\hat{u}_h, P_L \frac{\partial}{\partial y}\hat{u}_h \right)$$

つまり、

$$\nabla\hat{u}_h = \left(\frac{\partial}{\partial x}\hat{u}_h, \frac{\partial}{\partial y}\hat{u}_h \right) \in L^2 \times L^2,$$
$$\overline{\nabla}\hat{u}_h = \left(\overline{\nabla}_x\hat{u}_h, \overline{\nabla}_y\hat{u}_h \right) \in S_h^* \times S_h^*$$

とすると、

$$\left(\overline{\nabla}_x\hat{u}_h, \phi^* \right) = \left(\frac{\partial}{\partial x}\hat{u}_h, \phi^* \right), \quad \forall \phi^* \in S_h^*$$
$$\left(\overline{\nabla}_y\hat{u}_h, \phi^* \right) = \left(\frac{\partial}{\partial y}\hat{u}_h, \phi^* \right), \quad \forall \phi^* \in S_h^*$$

を満たす。そこで、$\hat{u}_h$ の係数ベクトルを $\vec{u}_h$、$\overline{\nabla}\hat{u}_h$ の係数ベクトルを $\vec{u}_x, \vec{u}_y$ と書くと、

$$\vec{u}_x = (L^*)^{-1} K_x \vec{u}_h \tag{5.9}$$
$$\vec{u}_y = (L^*)^{-1} K_y \vec{u}_h \tag{5.10}$$

によって $\overline{\nabla}\hat{u}_h$ が決まる。ここで、L^* は $M \times M$ 行列であり、K_x, K_y は $M \times N$ 行列で正方行列でないことに注意。その要素はそれぞれ

$$(L^*)_{ij} = (\phi_i^*, \phi_j^*) \qquad (i, j = 1, \cdots, M)$$
$$(K_x)_{ij} = (\phi_i^*, \partial_x\phi_j) \quad (i = 1, \cdots, M, j = 1, \cdots, N)$$
$$(K_y)_{ij} = (\phi_i^*, \partial_y\phi_j) \quad (i = 1, \cdots, M, j = 1, \cdots, N)$$

である。

また、

$$\overline{\Delta}\hat{u}_h \equiv \nabla \cdot \overline{\nabla}\hat{u}_h \in L^2$$

と定義すると、$\overline{\nabla}\hat{u}_h, \overline{\Delta}\hat{u}_h$ に関しての Green の公式

$$\left(\overline{\nabla}\hat{u}_h, \nabla\psi \right)_{L^2} + \left(\overline{\Delta}\hat{u}_h, \psi \right)_{L^2} = 0, \quad \forall \psi \in H_0^1$$

が成り立つ。これより、

$$(\nabla v_0, \nabla\psi)_{L^2} = \left(\overline{\nabla}\hat{u}_h - \nabla\hat{u}_h, \nabla\psi\right)_{L^2} + \left(\overline{\Delta}\hat{u}_h + f(\hat{u}_h), \psi\right)_{L^2}, \quad \forall\psi \in H_0^1$$

を得る。そこで、$\psi = v_0 - v_h$ ととって $(\nabla v_h, \nabla(v_0 - v_h))_{L^2} = 0$ なので、
$(\nabla v_0, \nabla(v_0 - v_h))_{L^2} = (\nabla(v_0 - v_h), \nabla(v_0 - v_h))_{L^2}$ であることに注意すると、

$$\|\nabla v_0\| \leq \|\overline{\nabla}\hat{u}_h - \nabla\hat{u}_h\| + C_0 h\|\overline{\Delta}\hat{u}_h + f(\hat{u}_h)\| + \|\nabla v_h\| \tag{5.11}$$

を得る。

$\overline{\nabla}\hat{u}_h$ が $\nabla\hat{u}_h$ を近似し、さらに u にある程度の滑らかさがあれば、$\overline{\Delta}\hat{u}_h$ は Δu を近似するので、(5.11) の右辺は h に関してのあるオーダーで 0 に近づくことが期待される。

特に、高次要素を用いて $\hat{u}_h$ を計算すれば、残差誤差 $\|\nabla v_0\|$ は著しい減少を図ることができる。

5.4 擬似 Newton 法

前節 4.2 節 (4.4) と同様に $K = (-\Delta_D)^{-1}$ を同次 Dirichlet 条件付き Laplacian の逆作用素として、

$$F(w) \equiv K(f(\hat{u}_h + v_0 + w) - f(\hat{u}_h))$$

と定めれば、残差方程式 (5.5) は

$$w = F(w) \tag{5.12}$$

と書ける。(5.12) は $(w_h, w^\perp) \in S_h \times S^\perp$ についての連立方程式

$$w_h = P_h F(w_h + w^\perp) \tag{5.13}$$
$$w^\perp = F(w_h + w^\perp) - P_h F(w_h + w^\perp) \tag{5.14}$$

として考えられる。このうちの無限次元部分 (5.14) の右辺は誤差評価式 (3.10) により、

$$\left\|\nabla\left(F(w_h + w^\perp) - P_h F(w_h + w^\perp)\right)\right\| \leq C_0 h\|f(\hat{u}_h + v_0 + w_h + w^\perp)\|$$

と評価できる。

したがって、(5.13) を Newton 型の反復が可能な形に変換する。つまり、(5.13) に同値なものとして、

$$w_h = N_h(w_h + w^\perp)$$

を考える。ただし、

$$N_h(w_h + w^\perp) \equiv w_h - \left[I - P_h A'(\hat{u}_h)\right]_h^{-1}\left(w_h - P_h(w_h + w^\perp)\right)$$

である。ここで、$A'(\hat{u}_h) \equiv Kf'(\hat{u}_h)$ で $f'(\hat{u}_h)$ は f の $\hat{u}_h$ における Fréchet 微分であり、$[I - P_h A'(\hat{u}_h)]_h^{-1}$ は $I - P_h A'(\hat{u}_h)|_{S_h}$ の逆写像である。

いま、

$$N_h(w) \equiv P_h w - \left[I - P_h A'(\hat{u}_h)\right]_h^{-1}(P_h w - P_h F(w))$$
$$T(w) \equiv N_h(w) + (I - P_h)F(w)$$

とすると、

$$w = T(w)$$

は、(5.13)、(5.14) と同値、すなわち、(5.12) と同値である。

一方、$w = T(w)$ は

$$T(w) = P_h T(w) + (I - P_h)T(w)$$

と書ける。ここで、

$$\begin{aligned}
P_h T(w) &= N_h(w) \\
&= P_h w - \left[I - P_h A'(\hat{u}_h)\right]_h^{-1} (P_h w - P_h F(w)) \\
&= \left[I - P_h A'(\hat{u}_h)\right]_h^{-1} \left(P_h F(w) - P_h A'(\hat{u}_h) P_h w\right) \\
&= \left[I - P_h A'(\hat{u}_h)\right]_h^{-1} P_h K \left(f(w + \hat{u}_h + v_0) - f(\hat{u}_h) - f'(\hat{u}_h) P_h w\right) \qquad (5.15) \\
(I - P_h)T(w) &= (I - P_h)F(w) \qquad (5.16)
\end{aligned}$$

である。

このとき、(5.15) より $P_h T(w) \in S_h$ は

$$\begin{aligned}
&(\nabla P_h T(w), \nabla \phi)_{L^2} - \left(f'(\hat{u}_h) P_h T(w), \phi\right)_{L^2} \\
&\quad = \left(f(\hat{u}_h + v_0 + w_h + w^{\perp}) - f(\hat{u}_h) - f'(\hat{u}_h) w_h, \phi\right)_{L^2}, \quad \forall \phi \in S_h \qquad (5.17)
\end{aligned}$$

を満たすことが分かる。

そして、無限次元部分 (5.16) は誤差評価式 (3.10) より、

$$\begin{aligned}
\|\nabla (I - P_h)T(w)\| &= \|\nabla (I - P_h)F(w)\| \\
&\leq C_0 h \|f(\hat{u}_h + v_0 + w) - f(\hat{u}_h)\| \qquad (5.18)
\end{aligned}$$

と評価できる。

5.5 候補者集合

Schauder の不動点定理を適用するには、

$$T(W) \subset W$$

を満たす $W \subset H_0^1$; 有界凸閉集合を見つければよい。

そこで、候補者集合としては

$$W = W_h + W^{\perp}$$

として

$$\begin{aligned}
W_h &\equiv \left\{w_h \in S_h \,;\, (w_h)_i \in W_i = [-\overline{W}_i, \overline{W}_i] \; (i = 1, \cdots, n)\right\} \\
W^{\perp} &\equiv \left\{w^{\perp} \in S_h^{\perp} \,;\, \|\nabla w^{\perp}\| \leq \alpha\right\}
\end{aligned}$$

の形のものを選ぶことにする。

5.6 検証条件

前節 4.4 節と同様に、検証条件 $T(W) \subset W$ が成り立つための十分条件として、

$$\begin{cases} P_h T(W) \subset W_h \\ (I - P_h)T(W) \subset W^{\perp} \end{cases} \qquad (5.19)$$

を得る。

5.7 具体的な数値例

ここでは次のような半線形楕円型偏微分方程式を元に、具体的な計算手順を述べる。

$$\begin{cases} -\Delta u = u^2, & \text{in } \Omega = (0,1)\times(0,1) \\ u = 0, & \text{on } \partial\Omega \end{cases} \tag{5.20}$$

5.7.1 有限要素法

$(0,1)$ 区間の x 方向を $n+1$ 等分し、等分点に以下のように番号をつける。

$$0 = x_0 < x_2 < x_4 < \cdots < x_{2n} < x_{2(n+1)} = 1$$

y 方向も $n+1$ 等分し、同様に等分点に番号をつける。$h = \frac{1}{n+1}$ とする。

さらに、要素 $(x_i, x_{i+2})\times(y_j, y_{j+2})$ の各辺に 2 等分点 x_{i+1}、y_{j+1} をとり、2 等分点を通り座標軸に平行な直線を考え、その交点 5 つを節点に追加する。

その節点上で 1 をとり、他の節点で 0 をとる区分的双 2 次関数 ϕ_{ij} を考える。つまり、

$$\phi_{ij}(x_k, y_l) = \begin{cases} 1 & (i = k \text{ かつ } j = l) \\ 0 & (\text{otherwise}) \end{cases} \qquad (i,j = 1,\cdots,2n+1)$$

を満たすものである。

具体的には次の式で与えられる。$(i,j = 2,4,\cdots,2n,2(n+1))$

$$\phi_{i-1j-1}(x,y) = \begin{cases} \dfrac{4}{h}(x-x_{i-1})\left\{1-\dfrac{1}{h}(x-x_{i-1})\right\} & \\ \quad\times\dfrac{4}{h}(y-y_{j-1})\left\{1-\dfrac{1}{h}(y-y_{j-1})\right\} & (x_{i-2}\le x\le x_i,\ y_{j-2}\le y\le y_j) \\ 0 & (\text{otherwise}) \end{cases}$$

$$\phi_{i-1j}(x,y) = \begin{cases} \dfrac{4}{h}(x-x_{i-1})\left\{1-\dfrac{1}{h}(x-x_{i-1})\right\} & \\ \quad\times\dfrac{1}{h}(y-y_{j-1})\left\{\dfrac{2}{h}(y-y_{j-1})-1\right\} & (x_{i-2}\le x\le x_i,\ y_{j-2}\le y\le y_j) \\ \dfrac{4}{h}(x-x_{i-1})\left\{1-\dfrac{1}{h}(x-x_{i-1})\right\} & \\ \quad\times\left\{\dfrac{1}{h}(y-y_j)-1\right\}\left\{\dfrac{2}{h}(y-y_j)-1\right\} & (x_{i-2}\le x\le x_i,\ y_j\le y\le y_{j+2}) \\ 0 & (\text{otherwise}) \end{cases}$$

$$\phi_{ij-1}(x,y) = \begin{cases} \dfrac{1}{h}(x-x_{i-1})\left\{\dfrac{2}{h}(x-x_{i-1})-1\right\} & \\ \quad\times\dfrac{4}{h}(y-y_{j-1})\left\{1-\dfrac{1}{h}(y-y_{j-1})\right\} & (x_{i-2}\le x\le x_i,\ y_{j-2}\le y\le y_j) \\ \left\{\dfrac{1}{h}(x-x_i)-1\right\}\left\{\dfrac{2}{h}(x-x_i)-1\right\} & \\ \quad\times\dfrac{4}{h}(y-y_{j-1})\left\{1-\dfrac{1}{h}(y-y_{j-1})\right\} & (x_i\le x\le x_{i+2},\ y_{j-2}\le y\le y_j) \\ 0 & (\text{otherwise}) \end{cases}$$

$$
\phi_{ij}(x,y) = \begin{cases}
\dfrac{1}{h}(x-x_{i-1})\left\{\dfrac{2}{h}(x-x_{i-1})-1\right\} & \\
\quad\times\dfrac{1}{h}(y-y_{j-1})\left\{\dfrac{2}{h}(y-y_{j-1})-1\right\} & (x_{i-2}\le x\le x_i,\ y_{j-2}\le y\le y_j)\\
\dfrac{1}{h}(x-x_{i-1})\left\{\dfrac{2}{h}(x-x_{i-1})-1\right\} & \\
\quad\times\left\{\dfrac{1}{h}(y-y_j)-1\right\}\left\{\dfrac{2}{h}(y-y_j)-1\right\} & (x_{i-2}\le x\le x_i,\ y_j\le y\le y_{j+2})\\
\left\{\dfrac{1}{h}(x-x_i)-1\right\}\left\{\dfrac{2}{h}(x-x_i)-1\right\} & \\
\quad\times\dfrac{1}{h}(y-y_{j-1})\left\{\dfrac{2}{h}(y-y_{j-1})-1\right\} & (x_i\le x\le x_{i+2},\ y_{j-2}\le y\le y_j)\\
\left\{\dfrac{1}{h}(x-x_i)-1\right\}\left\{\dfrac{2}{h}(x-x_i)-1\right\} & \\
\quad\times\left\{\dfrac{1}{h}(y-y_j)-1\right\}\left\{\dfrac{2}{h}(y-y_j)-1\right\} & (x_i\le x\le x_{i+2},\ y_j\le y\le y_{j+2})\\
0 & (\text{otherwise})
\end{cases}
$$

有限要素空間 S_h は ϕ_{ij} $(i,j=1,\cdots,2n+1)$ で張られるものを考える。したがって、$N=2n+1$ とおくと、S_h の次元は N^2 である。

代表要素 $e:(x_i,x_{i+2})\times(y_j,y_{j+2})$ 上の局所的な節点の番号付けを図 7 のとおりにする。

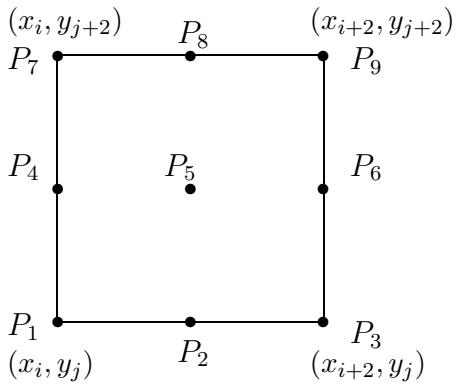


図 7: 代表要素 e の局所的番号 2

要素 e 上の基底関数 φ_i $(i=1,\cdots,9)$ は次を満たすようにとる。

$$\varphi_i(P_j)=\delta_{ij}\quad(i,j=1,\cdots,9)$$

具体的には次のようにとる。

$$
\begin{aligned}
\varphi_1(x,y) &= \left\{\frac{1}{h}(x-x_i)-1\right\}\left\{\frac{2}{h}(x-x_i)-1\right\}\left\{\frac{1}{h}(y-y_j)-1\right\}\left\{\frac{2}{h}(y-y_j)-1\right\}\\
\varphi_2(x,y) &= \frac{4}{h}(x-x_i)\left\{1-\frac{1}{h}(x-x_i)\right\}\left\{\frac{1}{h}(y-y_j)-1\right\}\left\{\frac{2}{h}(y-y_j)-1\right\}\\
\varphi_3(x,y) &= \frac{1}{h}(x-x_i)\left\{\frac{2}{h}(x-x_i)-1\right\}\left\{\frac{1}{h}(y-y_j)-1\right\}\left\{\frac{2}{h}(y-y_j)-1\right\}
\end{aligned}
$$

$$
\begin{aligned}
\varphi_4(x,y) &= \left\{\frac{1}{h}(x-x_i)-1\right\}\left\{\frac{2}{h}(x-x_i)-1\right\}\frac{4}{h}(y-y_j)\left\{1-\frac{1}{h}(y-y_j)\right\} \\
\varphi_5(x,y) &= \frac{4}{h}(x-x_i)\left\{1-\frac{1}{h}(x-x_i)\right\}\frac{4}{h}(y-y_j)\left\{1-\frac{1}{h}(y-y_j)\right\} \\
\varphi_6(x,y) &= \frac{1}{h}(x-x_i)\left\{\frac{2}{h}(x-x_i)-1\right\}\frac{4}{h}(y-y_j)\left\{1-\frac{1}{h}(y-y_j)\right\} \\
\varphi_7(x,y) &= \left\{\frac{1}{h}(x-x_i)-1\right\}\left\{\frac{2}{h}(x-x_i)-1\right\}\frac{1}{h}(y-y_j)\left\{\frac{2}{h}(y-y_j)-1\right\} \\
\varphi_8(x,y) &= \frac{4}{h}(x-x_i)\left\{1-\frac{1}{h}(x-x_i)\right\}\frac{1}{h}(y-y_j)\left\{\frac{2}{h}(y-y_j)-1\right\} \\
\varphi_9(x,y) &= \frac{1}{h}(x-x_i)\left\{\frac{2}{h}(x-x_i)-1\right\}\frac{1}{h}(y-y_j)\left\{\frac{2}{h}(y-y_j)-1\right\}
\end{aligned}
$$

問題 (5.20) の有限要素近似解は次式を満たす $u_h^* \in S_h$ であると定義される。

$$
(\nabla u_h^*, \nabla\phi)_{L^2} = \left((u_h^*)^2, \phi\right)_{L^2}, \quad \forall\phi \in S_h
$$

5.7.2 $\|\nabla v_h\|$ の評価

$v_h \in S_h$ は (5.6) を満たすものであった。再掲すると、

$$
(\nabla v_h, \nabla\phi)_{L^2} = \left(\hat{u}_h^2, \phi\right)_{L^2} - (\nabla\hat{u}_h, \nabla\phi)_{L^2}, \quad \forall\phi \in S_h
$$

である。これは、v_h の係数ベクトルを $\vec{v}_h$ で表すと、$\vec{v}_h$ に関する連立一次方程式

$$
D\vec{v}_h = L\overrightarrow{u^2} - D\vec{u}
$$

となる。ただし、$\hat{u}_h = \sum_{i,j=1}^{N} u_{ij}\phi_{ij}$ の係数ベクトルを $\vec{u}$、$\hat{u}_h^2$ を $\sum_{i,j=1}^{N} u_{ij}^2\phi_{ij}$ で近似し、その係数ベクトルを $\overrightarrow{u^2}$ と表す。

ここで、$D = (D_{ijkl}), L = (L_{ijkl})$ は $N^2 \times N^2$ 行列で、

$$
\begin{aligned}
D_{ijkl} &= (\nabla\phi_{ij}, \nabla\phi_{kl}) \\
L_{ijkl} &= (\phi_{ij}, \phi_{kl})
\end{aligned}
\tag{5.21}
$$

である。

この連立一次方程式を解き、得られた $\vec{v}_h$ を用いて、$\|\nabla v_h\|$ を

$$
\|\nabla v_h\| = \sqrt{\vec{v}_h{}^T\ D\ \vec{v}_h}
$$

と計算する。

また、$\|v_h\|$ も同様に

$$
\|v_h\| = \sqrt{\vec{v}_h{}^T\ L\ \vec{v}_h}
$$

と計算する。

5.7.3 $\|\nabla v_0\|$ の a posteriori 評価

S_h の基底に $\partial\Omega$ 上の節点に対する基底を加えたものを S_h^* とすると、S_h^* の次元は $M^2 \equiv (2n+1)^2 + 8(n+1) = (2(n+1)+1)^2$ である。

$\overline{\nabla}\hat{u}_h$ の定め方は、(5.9)、(5.10) より

$$\begin{aligned} \vec{u}_x &= (L^*)^{-1} K_x \vec{u}_h \\ \vec{u}_y &= (L^*)^{-1} K_y \vec{u}_h \end{aligned}$$

で定められる。ここで

$$\begin{aligned} (L^*)_{ijkl} &= \left(\phi_{ij}^*, \phi_{kl}^*\right)_{L^2} \quad (M^2 \times M^2 \text{行列}) \\ (K_x)_{ij} &= \left(\phi_{ij}^*, \frac{\partial}{\partial x}\phi_{kl}\right)_{L^2} \quad (M^2 \times N^2 \text{行列}) \\ (K_y)_{ij} &= \left(\phi_{ij}^*, \frac{\partial}{\partial y}\phi_{kl}\right)_{L^2} \quad (M^2 \times N^2 \text{行列}) \end{aligned} \tag{5.22}$$

である。

∇v_0 の具体的な計算は

$$\|\nabla v_0\| \le \|\overline{\nabla}\hat{u}_h - \nabla\hat{u}_h\| + C_0 h \|\overline{\Delta}\hat{u}_h + \hat{u}_h^2\| + \|\nabla v_h\|$$

を

$$\begin{aligned} &\|\overline{\nabla}\hat{u}_h - \nabla\hat{u}_h\|^2 \\ &= \int_\Omega \left(\overline{\nabla}\hat{u}_h - \nabla\hat{u}_h\right)\left(\overline{\nabla}\hat{u}_h - \nabla\hat{u}_h\right) \\ &= \int_\Omega \left\{ \left(\overline{\nabla}_x\hat{u}_h, \overline{\nabla}_y\hat{u}_h\right)\cdot\left(\overline{\nabla}_x\hat{u}_h, \overline{\nabla}_y\hat{u}_h\right) - 2\left(\overline{\nabla}_x\hat{u}_h, \overline{\nabla}_y\hat{u}_h\right)\cdot\left(\frac{\partial}{\partial x}\hat{u}_h, \frac{\partial}{\partial y}\hat{u}_h\right) + \nabla\hat{u}_h\cdot\nabla\hat{u}_h \right\} \\ &= \sum_{i,j,k,l=1}^{M} \overline{\nabla}_x u_{kl} \left(\phi_{kl}^*, \phi_{ij}^*\right)_{L^2} \overline{\nabla}_x u_{ij} + \sum_{i,j,k,l=1}^{M} \overline{\nabla}_y u_{kl} \left(\phi_{kl}^*, \phi_{ij}^*\right)_{L^2} \overline{\nabla}_y u_{ij} \\ &\quad - 2\sum_{i,j=1}^{M}\sum_{k,l=1}^{N} \overline{\nabla}_x u_{ij} \left(\phi_{ij}^*, \frac{\partial}{\partial x}\phi_{kl}\right)_{L^2} u_{kl} - 2\sum_{i,j=1}^{M}\sum_{k,l=1}^{N} \overline{\nabla}_y u_{ij} \left(\phi_{ij}^*, \frac{\partial}{\partial y}\phi_{kl}\right)_{L^2} u_{kl} \\ &\quad + \sum_{i,j,k,l=1}^{N} u_{ij} \left(\nabla\phi_{ij}, \nabla\phi_{kl}\right)_{L^2} u_{kl} \\ &= \left(\overline{\nabla}_x\vec{u} + \overline{\nabla}_y\vec{u}\right)^T L^* \left(\overline{\nabla}_x\vec{u} + \overline{\nabla}_y\vec{u}\right) - 2\overline{\nabla}_x\vec{u}^T\ K_x\ \vec{u} - 2\overline{\nabla}_y\vec{u}^T\ K_y\ \vec{u} + \vec{u}^T\ D\ \vec{u} \end{aligned}$$

$$\begin{aligned} &\|\overline{\Delta}\hat{u}_h + \hat{u}_h^2\|^2 \\ &= \int \left\{ \hat{u}_h^2\hat{u}_h^2 + 2\hat{u}_h^2\overline{\Delta}\hat{u}_h + \overline{\Delta}\hat{u}_h\cdot\overline{\Delta}\hat{u}_h \right\} \\ &= \sum_{i,j,k,l=1}^{N} u_{ij}^2 \left(\phi_{ij}, \phi_{kl}\right)_{L^2} u_{kl}^2 + 2\sum_{i,j=1}^{M}\sum_{k,l=1}^{N} \overline{\nabla}_x u_{ij} \left(\frac{\partial}{\partial x}\phi_{ij}^*, \phi_{kl}\right)_{L^2} u_{kl}^2 \\ &\quad + 2\sum_{i,j=1}^{M}\sum_{k,l=1}^{N} \overline{\nabla}_y u_{ij} \left(\frac{\partial}{\partial y}\phi_{ij}^*, \phi_{kl}\right)_{L^2} u_{kl}^2 + \sum_{i,j,k,l=1}^{M} \overline{\nabla}_x u_{ij} \left(\frac{\partial}{\partial x}\phi_{ij}^*, \frac{\partial}{\partial x}\phi_{kl}^*\right)_{L^2} \overline{\nabla}_x u_{kl} \\ &\quad + 2\sum_{i,j,k,l=1}^{M} \overline{\nabla}_y u_{ij} \left(\frac{\partial}{\partial y}\phi_{ij}^*, \frac{\partial}{\partial x}\phi_{kl}^*\right)_{L^2} \overline{\nabla}_x u_{kl} + \sum_{i,j,k,l=1}^{M} \overline{\nabla}_y u_{ij} \left(\frac{\partial}{\partial y}\phi_{ij}^*, \frac{\partial}{\partial y}\phi_{kl}^*\right)_{L^2} \overline{\nabla}_y u_{kl} \end{aligned}$$

$$= \overrightarrow{u^2}^\top \; L \; \overrightarrow{u^2} + 2\overline{\nabla}_x \vec{u}^\top \; \tilde{K}_x \; \overrightarrow{u^2} + 2\overline{\nabla}_y \vec{u}^\top \; \tilde{K}_y \; \overrightarrow{u^2}$$
$$+ \overline{\nabla}_x \vec{u}^\top \; D_{xx} \; \overline{\nabla}_x \vec{u} + 2\overline{\nabla}_y \vec{u}^\top \; D_{yx} \; \overline{\nabla}_x \vec{u} + \overline{\nabla}_y \vec{u}^\top \; D_{yy} \; \overline{\nabla}_y \vec{u}$$

となることを利用して計算する。

ここで、$L^*,\ K_x,\ K_y$ は (5.22) で、$D,\ L$ は (5.21) で表されたものであり、

$$\begin{aligned}
(\tilde{K}_x)_{ijkl} &= \left(\frac{\partial}{\partial x}\phi^*_{ij}, \phi_{kl}\right)_{L^2} \quad (M^2 \times N^2 \text{行列}) \\
(\tilde{K}_y)_{ijkl} &= \left(\frac{\partial}{\partial y}\phi^*_{ij}, \phi_{kl}\right)_{L^2} \quad (M^2 \times N^2 \text{行列}) \\
(D_{xx})_{ijkl} &= \left(\frac{\partial}{\partial x}\phi^*_{ij}, \frac{\partial}{\partial x}\phi^*_{kl}\right)_{L^2} \quad (M^2 \times M^2 \text{行列}) \\
(D_{yy})_{ijkl} &= \left(\frac{\partial}{\partial y}\phi^*_{ij}, \frac{\partial}{\partial y}\phi^*_{kl}\right)_{L^2} \quad (M^2 \times M^2 \text{行列}) \\
(D_{yx})_{ijkl} &= \left(\frac{\partial}{\partial y}\phi^*_{ij}, \frac{\partial}{\partial x}\phi^*_{kl}\right)_{L^2} \quad (M^2 \times M^2 \text{行列})
\end{aligned}$$

である。

代表要素 e においては、$(L_e)_{ij} \equiv \left(\varphi_i, \varphi^*_j\right)_{L^2(e)}$, $(K^x_e)_{ij} = (\tilde{K}^x_e)^T_{ij} \equiv \left(\varphi_i, \frac{\partial}{\partial x}\varphi_j\right)_{L^2(e)}$, $(K^y_e)_{ij} = (\tilde{K}^y_e)^T_{ij} \equiv \left(\varphi_i, \frac{\partial}{\partial y}\varphi_j\right)_{L^2(e)}$, $(D^{xx}_e)_{ij} \equiv \left(\frac{\partial}{\partial x}\varphi_i, \frac{\partial}{\partial x}\varphi_j,\right)_{L^2(e)}$, $(D^{yy}_e)_{ij} \equiv \left(\frac{\partial}{\partial y}\varphi_i, \frac{\partial}{\partial y}\varphi_j,\right)_{L^2(e)}$, $(D^{yx}_e)_{ij} \equiv \left(\frac{\partial}{\partial y}\varphi_i, \frac{\partial}{\partial x}\varphi_j,\right)_{L^2(e)}$ $(i,j = 1, \cdots, 9)$ はそれぞれ、

$$L_e = \frac{h^2}{900}\begin{pmatrix}
16 & 8 & -4 & 8 & 4 & -2 & -4 & -2 & 1 \\
 & 64 & 8 & 4 & 32 & 4 & -2 & -16 & -2 \\
 & & 16 & -2 & 4 & 8 & 1 & -2 & -4 \\
 & & & 64 & 32 & -16 & 8 & 4 & -2 \\
 & & & & 256 & 32 & 4 & 32 & 4 \\
 & & & & & 64 & -2 & 4 & 8 \\
 & & \text{Sym} & & & & 16 & 8 & -4 \\
 & & & & & & & 64 & 8 \\
 & & & & & & & & 16
\end{pmatrix}$$

$$K^x_e = \frac{h}{180}\begin{pmatrix}
-12 & 16 & -4 & -6 & 8 & -2 & 3 & -4 & 1 \\
-16 & 0 & 16 & -8 & 0 & 8 & 4 & 0 & -4 \\
4 & -16 & 12 & 2 & -8 & 6 & -1 & 4 & -3 \\
-6 & 8 & -2 & -48 & 64 & -16 & -6 & 8 & -2 \\
-8 & 0 & 8 & -64 & 0 & 64 & -8 & 0 & 8 \\
2 & -8 & 6 & 16 & -64 & 48 & 2 & -8 & 6 \\
3 & -4 & 1 & -6 & 8 & -2 & -12 & 16 & -4 \\
4 & 0 & -4 & -8 & 0 & 8 & -16 & 0 & 16 \\
-1 & 4 & -3 & 2 & -8 & 6 & 4 & -16 & 12
\end{pmatrix}$$

$$K_e^y = \frac{h}{180}\begin{pmatrix} -12 & -6 & 3 & 16 & 8 & -4 & -4 & -2 & 1 \\ -6 & 48 & -6 & 8 & 64 & 8 & -2 & -16 & -2 \\ 3 & -6 & -12 & -4 & 8 & 16 & 1 & -2 & -4 \\ -16 & -8 & 4 & 0 & 0 & 0 & 16 & 8 & -4 \\ -8 & -64 & -8 & 0 & 0 & 0 & 8 & 64 & 8 \\ 4 & -8 & -16 & 0 & 0 & 0 & -4 & 8 & 16 \\ 4 & 2 & -1 & -16 & -8 & 4 & 12 & 6 & -3 \\ 2 & 16 & 2 & -8 & -64 & -8 & 6 & 48 & 6 \\ -1 & 2 & 4 & 4 & -8 & -16 & -3 & 6 & 12 \end{pmatrix}$$

$$D_e^{xx} = \frac{1}{90}\begin{pmatrix} 28 & -32 & 4 & 14 & -16 & 2 & -7 & 8 & -1 \\ & 64 & -32 & -16 & 32 & -16 & 8 & -16 & 8 \\ & & 28 & 2 & -16 & 14 & -1 & 8 & -7 \\ & & & 112 & -128 & 16 & 14 & -16 & 2 \\ & & & & 256 & -128 & -16 & 32 & -16 \\ & & & & & 112 & 2 & -16 & 14 \\ & & \text{Sym} & & & & 28 & -32 & 4 \\ & & & & & & & 64 & -32 \\ & & & & & & & & 28 \end{pmatrix}$$

$$D_e^{yy} = \frac{1}{90}\begin{pmatrix} 28 & 14 & -7 & -32 & -16 & 8 & 4 & 2 & -1 \\ & 112 & 14 & -16 & -128 & -16 & 2 & 16 & 2 \\ & & 28 & 8 & -16 & -32 & -1 & 2 & 4 \\ & & & 64 & 32 & -16 & -32 & -16 & 8 \\ & & & & 256 & 32 & -16 & -128 & -16 \\ & & & & & 64 & 8 & -16 & -32 \\ & & \text{Sym} & & & & 28 & 14 & -7 \\ & & & & & & & 112 & 14 \\ & & & & & & & & 28 \end{pmatrix}$$

$$D_e^{yx} = \frac{1}{36}\begin{pmatrix} 9 & -12 & 3 & -12 & -16 & 4 & -3 & 4 & -1 \\ 12 & 0 & -12 & 16 & 0 & -16 & -4 & 0 & 4 \\ -3 & 12 & -9 & -4 & 16 & -12 & 1 & -4 & 3 \\ -12 & 16 & -4 & 0 & 0 & 0 & 12 & -16 & 4 \\ -16 & 0 & 16 & 0 & 0 & 0 & 16 & 0 & -16 \\ 4 & -16 & 12 & 0 & 0 & 0 & -4 & 16 & -12 \\ 3 & -4 & 1 & -12 & 16 & -4 & -9 & 12 & -3 \\ 4 & 0 & -4 & -16 & 0 & 16 & -12 & 0 & 12 \\ -1 & 4 & -3 & 4 & -16 & 12 & 3 & -12 & 9 \end{pmatrix}$$

である。

5.7.4 $T(w)$ の具体的な評価

$P_hT(w) \in S_h$ より $P_hT(w) = \sum_{i,j=1}^{N} T_{ij}\phi_{ij}$ と表すことができる。これを (5.17) に代入すれば、$N^2 \times N^2$ 行列 $G = (G_{ijkl})$ と N^2 次元ベクトル $\vec{g}_h = (g_{ij})$ をそれぞれ、

$$
\begin{aligned}
G_{ijkl} &= (\nabla\phi_{ij}, \nabla\phi_{kl})_{L^2} - (2\hat{u}_h\phi_{ij}, \phi_{kl})_{L^2} \\
g_{ij} &= \left((\hat{u}_h + v_0 + w_h + w^\perp)^2 - \hat{u}_h^2 - 2\hat{u}_h w_h, \phi_{ij}\right)_{L^2}
\end{aligned}
$$

と定めれば、

$$
G\vec{T}_h = \vec{g}_h
$$

が成り立つ。ここで、$\vec{T}_h \equiv (T_{ij})$ である。

ところが、v_0 などは値の範囲がわかるだけなので、$\vec{g}_h$ は区間値を持つベクトルとして扱わなければならない。そこで、過大評価となってしまうが、$\vec{\mathfrak{g}}_h = [-|\vec{g}_h|, |\vec{g}_h|]$ (絶対値は成分ごと) と置き換えて、区間演算の理論 (4.12) を適用すれば、

$$
\vec{T}_h \in |G^{-1}|\vec{\mathfrak{g}}_h \quad \text{(不等号は成分ごと)}
$$

によって $\vec{T}_h$ の値の範囲が定まることになる。

5.7.5 $|\vec{g}_h|$ の評価

まず、

$$
\begin{aligned}
g_{ij} &= \left((\hat{u}_h + v_0 + w_h + w^\perp)^2 - \hat{u}_h^2 - 2\hat{u}_h w_h, \phi_{ij}\right)_{L^2} \\
&= \left(w_h^2, \phi_{ij}\right)_{L^2} + 2\left((\hat{u}_h + w_h)(v_0 + w^\perp), \phi_{ij}\right)_{L^2} + \left((v_0 + w^\perp)^2, \phi_{ij}\right)_{L^2}
\end{aligned}
$$

を評価することにする。

第 1 項は

$$
\begin{aligned}
\left(w_h^2, \phi_{i'j'}\right)_{L^2} &= \left(\sum_{i,j=1}^{N} W_{ij}\phi_{ij} \sum_{k,l=1}^{N} W_{kl}\phi_{kl}, \phi_{i'j'}\right)_{L^2} \\
&= \sum_{i,j,k,l=1}^{N} \left(\phi_{i'j'}, \phi_{kl}\phi_{ij}\right)_{L^2} W_{ij}W_{kl}
\end{aligned}
$$

であるから、

$$
\left|\left(w_h^2, \phi_{i'j'}\right)_{L^2}\right| \le \sum_{i,j,k,l=1}^{N} \left|\left(\phi_{i'j'}, \phi_{kl}\phi_{ij}\right)_{L^2}\right| \overline{W}_{ij}\overline{W}_{kl}
$$

と評価できる。

第 2 項は、Schwarz の不等式と、$\|v_0\|$ に対する上からの評価 (5.8) と Poisson 方程式 (3.1) に対する誤差評価式 (3.11) より得られる

$$
\|w^\perp\| \le C_0 h \|\nabla w^\perp\| \tag{5.23}
$$

を用いて、

$$
\begin{aligned}
\left|\left(2(\hat{u}_h + w_h)(v_0 + w^\perp), \phi_{ij}\right)_{L^2}\right| &\le 2\|\hat{u}_h + w_h\|_{L^\infty}\|v_0 + w^\perp\|\|\phi_{ij}\| \\
&\le 2\|\hat{u}_h + w_h\|_{L^\infty}\left(C_0 h\|\nabla v_0\| + \|v_h\| + C_0 h\alpha\right)\|\phi_{ij}\|
\end{aligned}
$$

と評価する。

最後に第 3 項は、第 2 項と同様に (5.8) と (5.23) を用いて

$$\begin{aligned}\left|\left((v_0+w^\perp)^2,\phi_{ij}\right)_{L^2}\right| &\leq \|\phi_{ij}\|_{L^\infty}\|v_0+w^\perp\|^2 \\ &\leq \|\phi_{ij}\|_{L^\infty}\left(C_0h\|\nabla v_0\|+\|v_h\|+C_0h\alpha\right)^2\end{aligned}$$

と評価する。

ここで得られた評価を元に、$\vec{T}_h$ の値の範囲が得られる。なお、$\vec{\tilde{g}}_h$ の性質より、$\vec{T}_h$ の値の範囲は $\vec{T}_h \in [-|G^{-1}||\vec{g}_h|, |G^{-1}||\vec{g}_h|]$ となる。

5.7.6 無限次元部分の評価

次に無限次元部分 (5.18) の右辺の評価を行う。まず、(5.18) の右辺を次のように評価する。

$$\begin{aligned}&C_0h\|(\hat{u}_h+v_0+w_h+w^\perp)^2-\hat{u}_h^2\| \\ &\quad\leq C_0h\left\{\|(2\hat{u}_h+w_h)w_h\|+2\|(\hat{u}_h+w_h)(v_0+w^\perp)\|+\|(v_0+w^\perp)^2\|\right\}\end{aligned}$$

このとき、第 1 項は、

$$\|(2\hat{u}_h+w_h)w_h\| \leq \|w_h\|_{L^\infty}\|2\hat{u}_h+w_h\|$$

と評価できる。$\|2\hat{u}_h+w_h\|$ の評価は 4 節の具体的な数値例 4.6.2 節と同様に、

$$\|2\hat{u}_h+w_h\| = \sqrt{\sum_{i,j,k,l=1}^{N}(2u_{ij}+W_{ij})L_{ijkl}(2u_{kl}+W_{kl})}$$

と計算する。

また、第 2 項は (5.8) と (5.23) を用いて

$$\|(\hat{u}_h+w_h)(v_0+w^\perp)\| \leq \|\hat{u}_h+w_h\|_{L^\infty}\left(C_0h\|\nabla v_0\|+\|v_h\|+C_0h\alpha\right)$$

と評価する。

最後に、第 3 項は Schwarz の不等式から、

$$\|(v_0+w^\perp)^2\| \leq \sqrt{\|v_0+w^\perp\|\|v_0+w^\perp\|_{L^6}^3}$$

とする。

ただし、一般に実数 $p \geq 0$ に対し、

$$\|v\|_{L^p} \equiv \left(\int_\Omega |v|^p\right)^{1/p}$$

と定義される。

また、Sobolev の埋め込み定理 (Adams [1]) から次が成り立つ (Johnson [4])。

$$\|u\|_{L^p} \leq \frac{p}{4}|\Omega|^{1/p}\left\|\frac{\partial u}{\partial x}\right\|^{1/2}\left\|\frac{\partial u}{\partial y}\right\|^{1/2}, \quad \forall u \in H_0^1,\ 2 \geq p \in \mathbf{Z}$$

ここで、$|\Omega|$ は領域 Ω の面積を表す。これと、不等式

$$2\left\|\frac{\partial u}{\partial x}\right\|\left\|\frac{\partial u}{\partial y}\right\| \leq \|\nabla u\|^2$$

を用いると、

$$\|u\|_{L^p} \leq |\Omega|^{1/p} \frac{p\sqrt{2}}{8} \|\nabla u\|$$

となる。

したがって、$\|v_0 + w^\perp\|$ の評価に、今までと同様に (5.8), (5.23) を用いると

$$\|(v_0 + w^\perp)^2\| \leq \sqrt{(C_0 h \|\nabla v_0\| + \|v_h\| + C_0 h\alpha) \left(\frac{3}{\sqrt{8}}\right)^3 (\|\nabla v_0\| + \alpha)^3}$$

と評価されることになる。

これらより、

$$C_0 h \left\{ \|(2\hat{u}_h + w_h) w_h\| + \|(\hat{u}_h + w_h)(v_0 + w^\perp)\| + \|(v_0 + w^\perp)^2\| \right\} \leq T^\perp$$

となる算定可能な定数 $T^\perp$ を得ることができる。

5.7.7 具体的な検証条件とアルゴリズム

Schauder の不動点定理の条件を満足するための十分条件 (5.19) は次のようなものになる。

$$\begin{cases} \left(|G^{-1}||\vec{g}_h|\right)_{ij} \leq \overline{W}_{ij} \\ T^\perp \leq \alpha \end{cases} \tag{5.24}$$

これをもとにした具体的なアルゴリズムはほぼ 4.6.3 節と同じであるが、次のようなものになる。

Step.1 正定数 $\delta \ll 1$ を定める。

Step.2 はじめに $W_{ij}^{(0)} = [0, 0]$, $\alpha^{(0)} = 0$ ととる。これらによって定まる候補者集合を $W^{(0)}$ とする。

Step.3 $W_{ij}^{(k)}$, $\alpha^{(k)}$ および、これらによって定まる候補者集合 $W^{(k)}$ に対して、条件 (5.24) をチェックする。満足すれば反復を終了する。このとき求める解は $\hat{u}_h + v_0 + W^{(k)}$ の中に存在する。そうでなかったら、次のステップへ。

Step.4 $W_{ij}^{(k)}$, $\alpha^{(k)}$, $W^{(k)}$ と、これらから定まる $(|G^{-1}||\vec{g}_h|)^{(k)}$, $T^{\perp(k)}$ を用いて、

$$\begin{aligned} W_{ij}^{(k+1)} &= (1+\delta) \left(|G^{-1}||\vec{g}_h|\right)_{ij}^{(k)} \quad (i, j = 1, \cdots, N), \\ \alpha^{(k+1)} &= (1+\delta) T^{\perp(k)} \end{aligned}$$

によって、$W_{ij}^{(k+1)}$, $\alpha^{(k+1)}$ を定め、これから新しい候補者集合 $W^{(k+1)}$ を定義して、前のステップに戻る。

あらかじめ決めておいた反復回数の最大に達するまでに検証条件を満足する候補者集合が得られなければ、検証は失敗である。その場合には、h をより小さく取り直すか、あるいは、$P_h T(w)$ の係数ベクトルや無限次元部分 $\|\nabla (I - P_h) T(w)\|$ の評価法をもっとシャープに行うようにして、やり直すことになる。

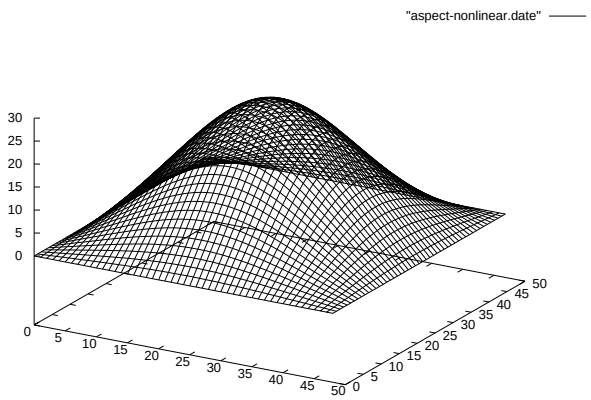


図 8: 問題 (5.20) の解の概形

5.7.8 数値計算結果

問題 (5.20) の解の概形を図 8 に示す。

まず、$\|\nabla v_0\|$ の評価についての実験を行った。これは 1 次要素のときの a priori 評価 (5.7)

$$\|\nabla v_0\| \leq C_0 h \|f(\hat{u}_h)\| + \|\nabla v_h\|$$

の第 1 項と a posteriori 評価 (5.11)

$$\|\nabla v_0\| \leq \|\overline{\nabla}\hat{u}_h - \nabla\hat{u}_h\| + C_0 h \|\overline{\Delta}\hat{u}_h + f(\hat{u}_h)\| + \|\nabla v_h\|$$

の第 1 項、第 2 項と、さらに、$\|\nabla v_h\|$ の値について表 5 のような結果を得た。また、2 次要素のときのそれぞれの値については表 6 のような結果を得た。ただし、計算にかかる手間を考慮して、2 次要素による計算は 1 次要素での計算に対して、その半分の分割数を対応させている。

また、それぞれの値はプログラム中では区間で計算しているが、ここでは区間の上限を表示するものとする。

分割数	$C_0 h\|f(\hat{u}_h)\|$	$\|\overline{\nabla}\hat{u}_h - \nabla\hat{u}_h\|$	$C_0 h\|\overline{\Delta}\hat{u}_h + f(\hat{u}_h)\|$	$\|\nabla v_h\|$	$\|\nabla v_0\|$
12	7.321834	4.878155	0.657466	2.895720e-08	5.535622
24	3.676465	2.455521	0.166690	6.713007e-08	2.622210
48	1.840283	1.229908	0.041819	1.452405e-07	1.271727

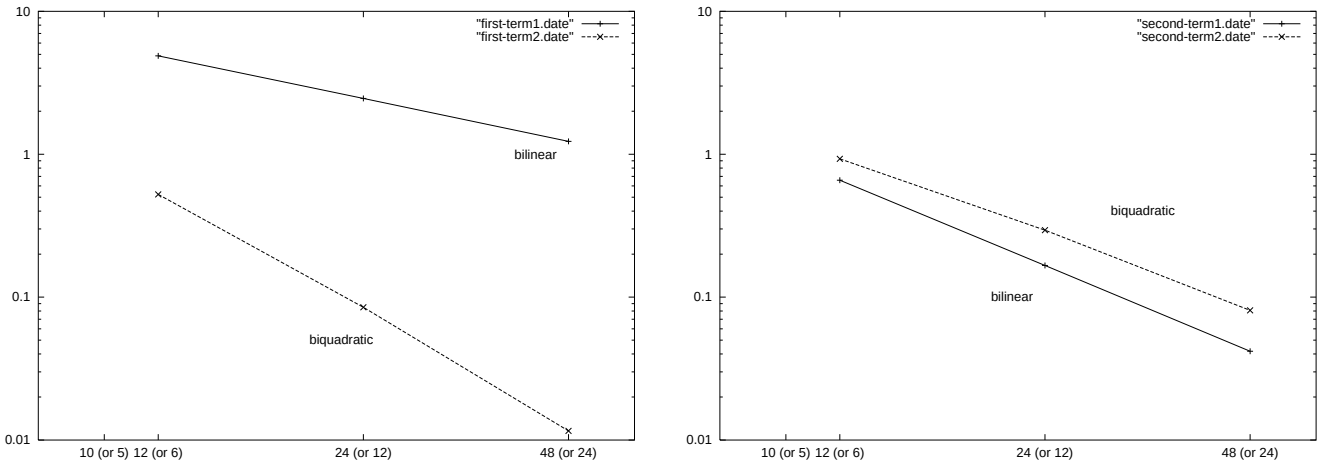
表 5: 1 次要素による $\|\nabla v_0\|$ の評価

分割数	$C_0 h\|f(\hat{u}_h)\|$	$\|\overline{\nabla}\hat{u}_h - \nabla\hat{u}_h\|$	$C_0 h\|\overline{\Delta}\hat{u}_h + f(\hat{u}_h)\|$	$\|\nabla v_h\|$	$\|\nabla v_0\|$
6	7.362464	0.523053	0.928331	6.624267e-11	1.451384
12	3.681906	0.084943	0.294242	1.606181e-10	0.379185
24	1.840976	0.011585	0.080753	7.328397e-10	0.092338

表 6: 2 次要素による $\|\nabla v_0\|$ の評価

これらの表より、$\|\nabla v_0\|$ の評価は、a priori 評価よりも、a posteriori 評価を用いたほうが、さらに、1 次要素よりも 2 次要素を用いたほうがよりよい結果を得ることが確認できた。また、$\|\nabla v_h\|$ の値もかなり小さくなっていることもあわせて確認できた。

次に、横軸を分割数、縦軸を 1 次要素、および、2 次要素での a posteriori 評価の第 1 項、第 2 項の値の両対数グラフをとると、次のようになる (図 9)。



(a) 分割数と $\|\overline{\nabla}\hat{u}_h - \nabla\hat{u}_h\|$ の両対数グラフ

(b) 分割数と $C_0 h\|\overline{\Delta}\hat{u}_h + f(\hat{u}_h)\|$ の両対数グラフ

図 9: 分割数と a posteriori 評価の両対数グラフ

図 9(a) において 1 次要素と 2 次要素で計算された $\|\overline{\nabla}\hat{u}_h - \nabla\hat{u}_h\|$ のグラフは直線状で、その傾きはそれぞれ

$$\frac{\log(1.229908) - \log(4.878155)}{\log(48) - \log(12)} = -0.993893 \approx -1.0$$
$$\frac{\log(0.011585) - \log(0.523053)}{\log(24) - \log(6)} = -2.748314 \approx -2.7$$

となり、1 次要素では h のオーダーで、2 次要素では h^2 以上のオーダーで減少していくのがわかる。

同様に、図 9(b) でも、$C_0 h\|\overline{\Delta}\hat{u}_h + f(\hat{u}_h)\|$ は 1 次要素、2 次要素ともにほぼ h^2 のオーダーで減少することがわかる。

このようにして得られた $\|\nabla v_0\|$ の評価を用いて、2 次要素で分割数が $6 \sim 24$ の場合について、問題 (5.20) の解の検証に成功した。表 7 にそのときの W の幅の最大と、α の値を示す。

分割数	反復回数	W の幅の最大	α
6	34	4.771164	1.938339
12	11	0.558071	0.102967
24	7	0.115933	0.010346

表 7: W と α

5.7.9 擬似 Newton 法による非強圧的線形楕円型偏微分方程式の解の検証

再び、問題 (4.16) を考える。問題 (4.16) とは

$$\begin{cases} -\Delta u = 20xyu + (2\pi - 1)\sin \pi x \sin \pi y, & \text{in } \Omega = (0,1) \times (0,1) \\ u = 0, & \text{on } \partial\Omega \end{cases}$$

であった。

逐次反復法を用いた検証 (前節 4.6.5) では u の係数 $20xy$ の値を大きくすると検証に失敗してしまうのであった。そこで、ここでは擬似 Newton 法を用いた反復によって解の検証を試みる。

その結果、分割数 20 で $-100xy \sim 100xy$ まで検証に成功した。そのときの、反復回数と W の幅、α の値を表 8 に示す。

係数	反復回数	W の幅の最大	α
-100	19	1.246430	0.542976
-80	15	1.124490	0.409322
-60	13	1.041823	0.302247
-40	10	0.986860	0.210498
-20	7	0.953508	0.126216
0	2	0.939234	0.042042
20	7	0.955760	0.138902
40	10	1.005269	0.284254
60	12	1.229139	0.844703
80	15	1.336969	0.868637
100	19	1.372122	0.754791

表 8: 擬似 Newton 法による検証

これより、逐次反復法が上手くいかない場合でも、擬似 Newton 法による反復を用いれば、解の検証に成功することも確認できた。

A　線形楕円型偏微分方程式に対する精度保証付き数値計算のプログラム

```
/*
 * varify.C
 *
 * -Δ u - π u = (2π - 1)sin π xsin π y in (0, 1) × (0, 1)
 * を精度保証付きで求める。
 *
 *
 * 入力値として
 *                分割数，総要素数，総節点数，境界条件を課す節点数，
 *                各要素を構成する節点番号，境界条件を課す節点番号
 *                近似解
 * が必要
 */

#include <iomanip.h>
#include "IntervalMatrix.h"
                          /*  区間行列、区間ベクトル、区間、行列、ベクトル  */
#include "IntegerVector.h"                                  /*  整数ベクトル  */
#include "Functions.h"               /*  ArcSin( ), Sin( ), Cos( ), Sqrt( )  */
#include "Utilities.h"                        /*  inverse( ) (逆行列の作成)  */

#define DELTA 10e-10                                      /*  delta-inflation  */
#define LOOP 50                                            /*  反復回数の最大  */
#define PI Constant::Pi                                      /*  厳密解用のπ  */

#define Div(x, y) DivBounds(x, y)
                         /*  実数同士の除算の結果を包み込むような区間で返す  */

INTERVAL Ih;                                                 /*  要素の辺長  */

struct Fournodes {                                         /*  局所節点番号表  */
  INT node[4];
};

/*  要素内で生成される行列の作成  */
void ecm(MATRIX & De, INTERVAL_MATRIX & Le)
{
  INT i, j;
  INTERVAL h2;

/*  (∇φ , ∇φ ) の計算  */
  for (i = 1; i <= 4; i++)
    De(i, i) = 2.0 / 3.0;
  De(1, 2) = De(2, 3) = De(3, 4) = De(2, 1) = De(3, 2) = De(4, 3) =
    De(1, 4) = De(4, 1) = -1.0 / 6.0;
  De(1, 3) = De(2, 4) = De(3, 1) = De(4, 2) = -1.0 / 3.0;

  h2 = Ih * Ih;

/*  (φ , φ ) の計算  */
  for (i = 1; i <= 4; i++)
    Le(i, i) = Div(1.0, 9.0) * h2;
  Le(1, 2) = Le(2, 3) = Le(3, 4) = Le(2, 1) = Le(3, 2) = Le(4, 3) =
    Le(1, 4) = Le(4, 1) = Div(1.0, 18.0) * h2;
  Le(1, 3) = Le(2, 4) = Le(3, 1) = Le(4, 2) = Div(1.0, 36.0) * h2;

}

/*  直接剛性法の実行  */
void assem(INT nelmt, MATRIX & D, INTERVAL_MATRIX & L, Fournodes *ielmt)
{
  INT i, j, ie, ii, jj;
  MATRIX De(4, 4);
  INTERVAL_MATRIX Le(4, 4);

  Clear(D);  Clear(L);

/* 直接剛性法 */
  ecm(De, Le);           /*  要素内で生成される行列は各要素毎には変わらない  */
  for (ie = 1; ie <= nelmt; ie++)
    for (i = 1; i <= 4; i++) {
```

```
      ii = ielmt[ie-1].node[i-1];
      for (j = 1; j <= 4; j++) {
jj = ielmt[ie-1].node[j-1];
D(ii, jj) += De(i, j);
L(ii, jj) += Le(i, j);
      }
    }
}


main(){
  INT i, j , k, ii;
  INT N;                                                    /*  分割数-1  */
  INT nnode;                                                /*  総節点数  */
  INT nelmt;                                                /*  総要素数  */
  INT nbc;                                          /*  境界条件を課す節点数  */



  cout << setprecision(13);

  cin >> N;
  cout << "分割数 : " << N + 1 << endl;

  Ih = Div(1.0, N + 1);

  cin >> nnode;
  cin >> nelmt;
  cin >> nbc;

  INTERVAL I, Pi;
  I = Hull(1.0);
  Pi = 2.0 * ArcSin(I);                                     /*  π（区間）  */

  Fournodes ielmt[nelmt];                          /*  各要素を構成する節点番号  */
  INTEGER_VECTOR ibc(nbc);                          /*  境界条件を課す節点番号  */

  for (i = 0; i < nelmt; i++)
    for (j = 0; j < 4; j++)
      cin >> ielmt[i].node[j];

  cin >> ibc;

  VECTOR u(nnode);                                          /*  近似解  */
  cin >> u;

  MATRIX D(nnode, nnode);
  INTERVAL_MATRIX L(nnode, nnode);

/*  直接剛性法による計算  */
  assem(nelmt, D, L, ielmt);


/*  Dirichlet 境界条件の考慮  */
  for (i = 1; i <= nbc; i++) {
    ii = ibc(i);
    for (j = 1; j <= nnode; j++) {
      D(ii, j) = D(j, ii) = 0.0;
      L(ii, j) = L(j , ii) = 0.0;
    }
    L(ii, ii) = 1.0;
    D(ii, ii) = 1.0;                                /*  逆行列を計算するため  */
  }


  MATRIX Dinv(nnode, nnode), Dia(nnode, nnode);

  Dinv = Inverse(D);                                        /*  D の逆行列  */

/*  D の逆行列の絶対値行列  */
  for (i = 1; i <= nnode; i++)
    for (j = 1; j <= nnode; j++)
      Dia(i, j) = Abs(Dinv(i, j));
```

```
/*  厳密解の計算（これは区間では計算していない） */
  REAL h;
  h = 1.0 / (N + 1);
  VECTOR x(nnode), y(nnode), Exact(nnode);

  k = 1;
  for (i = 0; i <= N+1; i++)
    for (j = 0; j <= N+1; j++) {
      x(k) = j * h;
      y(k) = i * h;
      k++;
    }
  for (i = 1; i <= nnode; i++)
    Exact(i) = 1.0 / PI * Sin(PI * x(i)) * Sin(PI * y(i));
  for (i = 1; i <= nbc; i++) {
    ii = ibc(i);
    Exact(ii) = 0.0;
  }



/*  主要部分  */
  INT count;                                                  /*  反復回数  */

  INTERVAL sin_norm;                      /*  ‖ sin π xsin π y‖  を区間で計算  */
  INTERVAL Ialp;                                /*  α ^(i) を区間で計算  */
  INTERVAL F_ort;                                             /*  α ^(i+1)  */

  INTERVAL_VECTOR Phi(nnode);              /*  ‖ φ _i‖ をベクトル形にする  */
  INTERVAL_VECTOR C(nnode);
  INTERVAL_VECTOR Y(nnode);                                   /*  |D^(-1)| C  */
  INTERVAL_VECTOR W(nnode);
  INTERVAL_VECTOR U(nnode);                                    /*  U = u + W  */

  REAL sup_norm;                                  /*  sup_(v_h ∈ U_h)‖ v_h‖  */

  REAL width, WIDTH;

  sin_norm = Div(1.0, 2.0);


/*  ‖ φ _i‖  の計算  */
  for (i = 1; i <= nnode; i++)
    Phi(i) = Sqrt(L(i,i));
  for (i = 1; i <= nbc; i++) {
    ii = ibc(i);
    Phi(ii) = 0.0;
  }


/*  反復の初期値  */
  Clear(W);
  Ialp = Hull(0.0);

/*  反復開始  */
  for (count = 1; count <= LOOP; count++) {

/*  |b_i| の評価  */
    C = Pi * (Abs(L) * Sup(W)) + Ih * Ialp * Phi;

    C = SymHull(Abs(C));                        /*  C_i = [ -|b_i|, |b_i|]  */
    Y = Dia * C;                                        /*  Y = |D^(-1)| C  */

    U = u + W;
    sup_norm = Abs(Sqrt(U * (L * U)));           /*  sup_(v_h ∈ U_h)‖ v_h‖  */

/*  F^⊥ の評価  */
    F_ort = Ih / Pi * (Pi * sup_norm + Ih * Ialp
        + (2.0 * Pi - 1.0) * sin_norm
       );
```

```
/* 検証条件  */
    if (Y <= W && Sup(F_ort) <= Inf(Ialp)) {
      cout << "Iteration Stop --- 反復回数 " << count << " 回" << endl;
      cout << endl;
      cout << "節点\t\tU(i)\t\t 厳密解\t\tU(i) の幅" << endl;

      WIDTH = 0.0;
      for (i = 1; i <= nnode; i++) {
cout << i << "  " << U(i) << "   " << Exact(i)
  << "   " << Diam(W(i)) << "  ";
if (Hull(Exact(i)) <= U(i))
  cout <<"  ◎ " << endl;
else
  cout <<"  ● " << endl;

width = Diam(W(i));
if (WIDTH < width)
  WIDTH = width;

      }
      cout << endl;

      cout << "区間 W_i の幅の最大 = " << WIDTH << endl;
      cout << "無限次元部分の半径 ; α  = " << Sup(Ialp) << endl;
      exit(0);
    }

    W = (1.0 + DELTA) * Y;
    Ialp = (1.0 + DELTA) * F_ort;
  }
  cout << "検証失敗" << endl;
  exit(0);
}
```

このプログラムの入力で必要な 分割数、総節点数、総要素数、境界条件を課す節点数、各要素を構成する節点番号、境界条件を課す節点番号は次のようなプログラムで出力される。

```
/*
 * input.c
 *
 */

#include <stdio.h>
#include <math.h>

main()
{
    int i, j, n;
    int node1, node2, node3, node4;
    double h;

    /* 分割数 n の入力 */
    scanf("%d", &n);

    h = 1.0 / (n + 1);

    /* 分割数-1, 総節点数, 総要素数, ディリクレ境界条件を課す節点数を表示 */
    printf("%d %d  %d  %d\n", n, (n + 2) * (n + 2), (n + 1) * (n + 1),
   4 * (n + 1));

    /* 各要素に属する節点番号を表示 */
    for (j = 1; j <= n+1; j++)
      for (i = 1; i <= n+1; i++){
node1 = i + (n + 2) * (j - 1);
node2 = node1 + 1;
node3 = i + (n + 2) * j + 1;
node4 = node3 - 1;

printf("%d  %d  %d  %d\n", node1, node2, node3, node4);
        }
    printf("\n");

    /* ディリクレ境界条件を課す節点番号を表示  */
```

```
    for (j = 0; j <= n+1; j++)
      printf("%d ", j+1);
    for (i = 1; i <= n; i++)
      printf("%d %d ", (n + 2) * i+1, (n + 2) * i + (n + 1)+1);
    for (j = 0; j <= n+1; j++)
      printf("%d ", (n + 2) * (n + 1) + j+1);
    printf("\n");

    return 0;
}
```

B　半線形楕円型偏微分方程式に対する精度保証付き数値計算のプログラム

```
/*
 * biquadratic.C --- 基底関数は 2 次
 *
 * -Δ u = u^2  in Ω = (0, 1) × (0, 1)
 *    u = 0    on ∂Ω
 *
 * の解を Newton-like 反復によって精度保証付きで求める。
 *
 * 入力値として
 *               分割数，総要素数，総節点数，境界条件を課す節点数，
 *               各要素を構成する節点番号，境界条件を課す節点番号
 *               近似解
 * が必要
 */

#include <iomanip.h>
#include "IntervalMatrix.h"
                          /*  区間行列、区間ベクトル、区間、行列、ベクトル  */
#include "IntegerVector.h"                                  /*  整数ベクトル  */
#include "Functions.h"                             /*  Sqrt( ), Constant::Pi  */
#include "Utilities.h"                         /*  Inverse( ) (逆行列の作成)  */
#include "LSS.h"
                            /*  Rump の方法にて連立一次方程式の解を求める  */

#define DELTA 1.0e-3                                       /*  Delta-inflation  */
#define LOOP 50                                             /*  反復回数の最大  */


#define Div(x, y) DivBounds(x, y)
                         /*  実数同士の除算の結果を包み込むような区間で返す  */
#define Mul(x, y) MulBounds(x, y)
                         /*  実数同士の乗算の結果を包み込むような区間で返す  */

struct Ninenodes {                                          /*  局所節点番号表  */
  INT node[9];
};

INTERVAL Ih;                                                  /*  要素の辺長  */
REAL Pi = Constant::Pi;


/*  要素ごとには変わらない行列などを計算  */
void ecm(INTERVAL_MATRIX & Le, INTERVAL_MATRIX & K_xe, INTERVAL_MATRIX & K_ye,
 INTERVAL_MATRIX & D_xxe, INTERVAL_MATRIX & D_yye,
 INTERVAL_MATRIX & D_yxe, INTERVAL (*Ae)[9][9])
{
  INT i, j, k;

  INTERVAL h2 = Ih * Ih;


/*  (φ , φ ) の計算  */
  Le(1, 1)            = Div(  4.0, 225.0) * h2;
  Le(1, 2) = Le(2, 1) = Div(  2.0, 225.0) * h2;
  Le(1, 3) = Le(3, 1) = Div(- 1.0, 225.0) * h2;
  Le(1, 4) = Le(4, 1) = Div(  2.0, 225.0) * h2;
  Le(1, 5) = Le(5, 1) = Div(  1.0, 225.0) * h2;
  Le(1, 6) = Le(6, 1) = Div(- 1.0, 450.0) * h2;
  Le(1, 7) = Le(7, 1) = Div(- 1.0, 225.0) * h2;
  Le(1, 8) = Le(8, 1) = Div(- 1.0, 450.0) * h2;
  Le(1, 9) = Le(9, 1) = Div(  1.0, 900.0) * h2;

  Le(2, 2)            = Div( 16.0, 225.0) * h2;
  Le(2, 3) = Le(3, 2) = Div(  2.0, 225.0) * h2;
  Le(2, 4) = Le(4, 2) = Div(  1.0, 225.0) * h2;
  Le(2, 5) = Le(5, 2) = Div(  8.0, 225.0) * h2;
  Le(2, 6) = Le(6, 2) = Div(  1.0, 225.0) * h2;
  Le(2, 7) = Le(7, 2) = Div(- 1.0, 450.0) * h2;
  Le(2, 8) = Le(8, 2) = Div(- 4.0, 225.0) * h2;
  Le(2, 9) = Le(9, 2) = Div(- 1.0, 450.0) * h2;
```

```
  Le(3, 3)            = Div(  4.0, 225.0) * h2;
  Le(3, 4) = Le(4, 3) = Div(- 1.0, 450.0) * h2;
  Le(3, 5) = Le(5, 3) = Div(  1.0, 225.0) * h2;
  Le(3, 6) = Le(6, 3) = Div(  2.0, 225.0) * h2;
  Le(3, 7) = Le(7, 3) = Div(  1.0, 900.0) * h2;
  Le(3, 8) = Le(8, 3) = Div(- 1.0, 450.0) * h2;
  Le(3, 9) = Le(9, 3) = Div(- 1.0, 225.0) * h2;

  Le(4, 4)            = Div( 16.0, 225.0) * h2;
  Le(4, 5) = Le(5, 4) = Div(  8.0, 225.0) * h2;
  Le(4, 6) = Le(6, 4) = Div(- 4.0, 225.0) * h2;
  Le(4, 7) = Le(7, 4) = Div(  2.0, 225.0) * h2;
  Le(4, 8) = Le(8, 4) = Div(  1.0, 225.0) * h2;
  Le(4, 9) = Le(9, 4) = Div(- 1.0, 450.0) * h2;

  Le(5, 5)            = Div( 64.0, 225.0) * h2;
  Le(5, 6) = Le(6, 5) = Div(  8.0, 225.0) * h2;
  Le(5, 7) = Le(7, 5) = Div(  1.0, 225.0) * h2;
  Le(5, 8) = Le(8, 5) = Div(  8.0, 225.0) * h2;
  Le(5, 9) = Le(9, 5) = Div(  1.0, 225.0) * h2;

  Le(6, 6)            = Div( 16.0, 225.0) * h2;
  Le(6, 7) = Le(7, 6) = Div(- 1.0, 450.0) * h2;
  Le(6, 8) = Le(8, 6) = Div(  1.0, 225.0) * h2;
  Le(6, 9) = Le(9, 6) = Div(  2.0, 225.0) * h2;

  Le(7, 7)            = Div(  4.0, 225.0) * h2;
  Le(7, 8) = Le(8, 7) = Div(  2.0, 225.0) * h2;
  Le(7, 9) = Le(9, 7) = Div(- 1.0, 225.0) * h2;

  Le(8, 8)            = Div( 16.0, 225.0) * h2;
  Le(8, 9) = Le(9, 8) = Div(  2.0, 225.0) * h2;

  Le(9, 9)            = Div(  4.0, 225.0) * h2;


/*  (φ , ∂_x φ ) の計算  */
  K_xe(1, 1) = Div(- 1.0, 15.0) * Ih;
  K_xe(1, 2) = Div(  4.0, 45.0) * Ih;
  K_xe(1, 3) = Div(- 1.0, 45.0) * Ih;
  K_xe(1, 4) = Div(- 1.0, 30.0) * Ih;
  K_xe(1, 5) = Div(  2.0, 45.0) * Ih;
  K_xe(1, 6) = Div(- 1.0, 90.0) * Ih;
  K_xe(1, 7) = Div(  1.0, 60.0) * Ih;
  K_xe(1, 8) = Div(- 1.0, 45.0) * Ih;
  K_xe(1, 9) = Div(  1.0,180.0) * Ih;

  K_xe(2, 1) = Div(- 4.0, 45.0) * Ih;
  K_xe(2, 2) = 0.0;
  K_xe(2, 3) = Div(  4.0, 45.0) * Ih;
  K_xe(2, 4) = Div(- 2.0, 45.0) * Ih;
  K_xe(2, 5) = 0.0;
  K_xe(2, 6) = Div(  2.0, 45.0) * Ih;
  K_xe(2, 7) = Div(  1.0, 45.0) * Ih;
  K_xe(2, 8) = 0.0;
  K_xe(2, 9) = Div(- 1.0, 45.0) * Ih;

  K_xe(3, 1) = Div(  1.0, 45.0) * Ih;
  K_xe(3, 2) = Div(- 4.0, 45.0) * Ih;
  K_xe(3, 3) = Div(  1.0, 15.0) * Ih;
  K_xe(3, 4) = Div(  1.0, 90.0) * Ih;
  K_xe(3, 5) = Div(- 2.0, 45.0) * Ih;
  K_xe(3, 6) = Div(  1.0, 30.0) * Ih;
  K_xe(3, 7) = Div(- 1.0,180.0) * Ih;
  K_xe(3, 8) = Div(  1.0, 45.0) * Ih;
  K_xe(3, 9) = Div(- 1.0, 60.0) * Ih;

  K_xe(4, 1) = Div(- 1.0, 30.0) * Ih;
  K_xe(4, 2) = Div(  2.0, 45.0) * Ih;
  K_xe(4, 3) = Div(- 1.0, 90.0) * Ih;
  K_xe(4, 4) = Div(- 4.0, 15.0) * Ih;
  K_xe(4, 5) = Div( 16.0, 45.0) * Ih;
  K_xe(4, 6) = Div(- 4.0, 45.0) * Ih;
```

```
  K_xe(4, 7) = Div(- 1.0, 30.0) * Ih;
  K_xe(4, 8) = Div(  2.0, 45.0) * Ih;
  K_xe(4, 9) = Div(- 1.0, 90.0) * Ih;

  K_xe(5, 1) = Div(- 2.0, 45.0) * Ih;
  K_xe(5, 2) = 0.0;
  K_xe(5, 3) = Div(  2.0, 45.0) * Ih;
  K_xe(5, 4) = Div(-16.0, 45.0) * Ih;
  K_xe(5, 5) = 0.0;
  K_xe(5, 6) = Div( 16.0, 45.0) * Ih;
  K_xe(5, 7) = Div(- 2.0, 45.0) * Ih;
  K_xe(5, 8) = 0.0;
  K_xe(5, 9) = Div(  2.0, 45.0) * Ih;

  K_xe(6, 1) = Div(  1.0, 90.0) * Ih;
  K_xe(6, 2) = Div(- 2.0, 45.0) * Ih;
  K_xe(6, 3) = Div(  1.0, 30.0) * Ih;
  K_xe(6, 4) = Div(  4.0, 45.0) * Ih;
  K_xe(6, 5) = Div(-16.0, 45.0) * Ih;
  K_xe(6, 6) = Div(  4.0, 15.0) * Ih;
  K_xe(6, 7) = Div(  1.0, 90.0) * Ih;
  K_xe(6, 8) = Div(- 2.0, 45.0) * Ih;
  K_xe(6, 9) = Div(  1.0, 30.0) * Ih;

  K_xe(7, 1) = Div(  1.0, 60.0) * Ih;
  K_xe(7, 2) = Div(- 1.0, 45.0) * Ih;
  K_xe(7, 3) = Div(  1.0,180.0) * Ih;
  K_xe(7, 4) = Div(- 1.0, 30.0) * Ih;
  K_xe(7, 5) = Div(  2.0, 45.0) * Ih;
  K_xe(7, 6) = Div(- 1.0, 90.0) * Ih;
  K_xe(7, 7) = Div(- 1.0, 15.0) * Ih;
  K_xe(7, 8) = Div(  4.0, 45.0) * Ih;
  K_xe(7, 9) = Div(- 1.0, 45.0) * Ih;

  K_xe(8, 1) = Div(  1.0, 45.0) * Ih;
  K_xe(8, 2) = 0.0;
  K_xe(8, 3) = Div(- 1.0, 45.0) * Ih;
  K_xe(8, 4) = Div(- 2.0, 45.0) * Ih;
  K_xe(8, 5) = 0.0;
  K_xe(8, 6) = Div(  2.0, 45.0) * Ih;
  K_xe(8, 7) = Div(- 4.0, 45.0) * Ih;
  K_xe(8, 8) = 0.0;
  K_xe(8, 9) = Div(  4.0, 45.0) * Ih;

  K_xe(9, 1) = Div(- 1.0,180.0) * Ih;
  K_xe(9, 2) = Div(  1.0, 45.0) * Ih;
  K_xe(9, 3) = Div(- 1.0, 60.0) * Ih;
  K_xe(9, 4) = Div(  1.0, 90.0) * Ih;
  K_xe(9, 5) = Div(- 2.0, 45.0) * Ih;
  K_xe(9, 6) = Div(  1.0, 30.0) * Ih;
  K_xe(9, 7) = Div(  1.0, 45.0) * Ih;
  K_xe(9, 8) = Div(- 4.0, 45.0) * Ih;
  K_xe(9, 9) = Div(  1.0, 15.0) * Ih;


/*  (φ , ∂_y φ ) の計算  */
  K_ye(1, 1) = Div(- 1.0, 15.0) * Ih;
  K_ye(1, 2) = Div(- 1.0, 30.0) * Ih;
  K_ye(1, 3) = Div(  1.0, 60.0) * Ih;
  K_ye(1, 4) = Div(  4.0, 45.0) * Ih;
  K_ye(1, 5) = Div(  2.0, 45.0) * Ih;
  K_ye(1, 6) = Div(- 1.0, 45.0) * Ih;
  K_ye(1, 7) = Div(- 1.0, 45.0) * Ih;
  K_ye(1, 8) = Div(- 1.0, 90.0) * Ih;
  K_ye(1, 9) = Div(  1.0,180.0) * Ih;

  K_ye(2, 1) = Div(- 1.0, 30.0) * Ih;
  K_ye(2, 2) = Div(- 4.0, 15.0) * Ih;
  K_ye(2, 3) = Div(- 1.0, 30.0) * Ih;
  K_ye(2, 4) = Div(  2.0, 45.0) * Ih;
  K_ye(2, 5) = Div( 16.0, 45.0) * Ih;
  K_ye(2, 6) = Div(  2.0, 45.0) * Ih;
  K_ye(2, 7) = Div(- 1.0, 90.0) * Ih;
```

```
K_ye(2, 8) = Div(- 4.0, 45.0) * Ih;
K_ye(2, 9) = Div(- 1.0, 90.0) * Ih;

K_ye(3, 1) = Div(  1.0, 60.0) * Ih;
K_ye(3, 2) = Div(- 1.0, 30.0) * Ih;
K_ye(3, 3) = Div(- 1.0, 15.0) * Ih;
K_ye(3, 4) = Div(- 1.0, 45.0) * Ih;
K_ye(3, 5) = Div(  2.0, 45.0) * Ih;
K_ye(3, 6) = Div(  4.0, 45.0) * Ih;
K_ye(3, 7) = Div(  1.0,180.0) * Ih;
K_ye(3, 8) = Div(- 1.0, 90.0) * Ih;
K_ye(3, 9) = Div(- 1.0, 45.0) * Ih;

K_ye(4, 1) = Div(- 4.0, 45.0) * Ih;
K_ye(4, 2) = Div(- 2.0, 45.0) * Ih;
K_ye(4, 3) = Div(  1.0, 45.0) * Ih;
K_ye(4, 4) = 0.0;
K_ye(4, 5) = 0.0;
K_ye(4, 6) = 0.0;
K_ye(4, 7) = Div(  4.0, 45.0) * Ih;
K_ye(4, 8) = Div(  2.0, 45.0) * Ih;
K_ye(4, 9) = Div(- 1.0, 45.0) * Ih;

K_ye(5, 1) = Div(- 2.0, 45.0) * Ih;
K_ye(5, 2) = Div(-16.0, 45.0) * Ih;
K_ye(5, 3) = Div(- 2.0, 45.0) * Ih;
K_ye(5, 4) = 0.0;
K_ye(5, 5) = 0.0;
K_ye(5, 6) = 0.0;
K_ye(5, 7) = Div(  2.0, 45.0) * Ih;
K_ye(5, 8) = Div( 16.0, 45.0) * Ih;
K_ye(5, 9) = Div(  2.0, 45.0) * Ih;

K_ye(6, 1) = Div(  1.0, 45.0) * Ih;
K_ye(6, 2) = Div(- 2.0, 45.0) * Ih;
K_ye(6, 3) = Div(- 4.0, 45.0) * Ih;
K_ye(6, 4) = 0.0;
K_ye(6, 5) = 0.0;
K_ye(6, 6) = 0.0;
K_ye(6, 7) = Div(- 1.0, 45.0) * Ih;
K_ye(6, 8) = Div(  2.0, 45.0) * Ih;
K_ye(6, 9) = Div(  4.0, 45.0) * Ih;

K_ye(7, 1) = Div(  1.0, 45.0) * Ih;
K_ye(7, 2) = Div(  1.0, 90.0) * Ih;
K_ye(7, 3) = Div(- 1.0,180.0) * Ih;
K_ye(7, 4) = Div(- 4.0, 45.0) * Ih;
K_ye(7, 5) = Div(- 2.0, 45.0) * Ih;
K_ye(7, 6) = Div(  1.0, 45.0) * Ih;
K_ye(7, 7) = Div(  1.0, 15.0) * Ih;
K_ye(7, 8) = Div(  1.0, 30.0) * Ih;
K_ye(7, 9) = Div(- 1.0, 60.0) * Ih;

K_ye(8, 1) = Div(  1.0, 90.0) * Ih;
K_ye(8, 2) = Div(  4.0, 45.0) * Ih;
K_ye(8, 3) = Div(  1.0, 90.0) * Ih;
K_ye(8, 4) = Div(- 2.0, 45.0) * Ih;
K_ye(8, 5) = Div(-16.0, 45.0) * Ih;
K_ye(8, 6) = Div(- 2.0, 45.0) * Ih;
K_ye(8, 7) = Div(  1.0, 30.0) * Ih;
K_ye(8, 8) = Div(  4.0, 15.0) * Ih;
K_ye(8, 9) = Div(  1.0, 30.0) * Ih;

K_ye(9, 1) = Div(- 1.0,180.0) * Ih;
K_ye(9, 2) = Div(  1.0, 90.0) * Ih;
K_ye(9, 3) = Div(  1.0, 45.0) * Ih;
K_ye(9, 4) = Div(  1.0, 45.0) * Ih;
K_ye(9, 5) = Div(- 2.0, 45.0) * Ih;
K_ye(9, 6) = Div(- 4.0, 45.0) * Ih;
K_ye(9, 7) = Div(- 1.0, 60.0) * Ih;
K_ye(9, 8) = Div(  1.0, 30.0) * Ih;
K_ye(9, 9) = Div(  1.0, 15.0) * Ih;
```

```
/*  (∂_xφ , ∂_xφ) の計算  */
  D_xxe(1, 1)               = Div( 14.0, 45.0);
  D_xxe(1, 2) = D_xxe(2, 1) = Div(-16.0, 45.0);
  D_xxe(1, 3) = D_xxe(3, 1) = Div(  2.0, 45.0);
  D_xxe(1, 4) = D_xxe(4, 1) = Div(  7.0, 45.0);
  D_xxe(1, 5) = D_xxe(5, 1) = Div(- 8.0, 45.0);
  D_xxe(1, 6) = D_xxe(6, 1) = Div(  1.0, 45.0);
  D_xxe(1, 7) = D_xxe(7, 1) = Div(- 7.0, 90.0);
  D_xxe(1, 8) = D_xxe(8, 1) = Div(  4.0, 45.0);
  D_xxe(1, 9) = D_xxe(9, 1) = Div(- 1.0, 90.0);

  D_xxe(2, 2)               = Div( 32.0, 45.0);
  D_xxe(2, 3) = D_xxe(3, 2) = Div(-16.0, 45.0);
  D_xxe(2, 4) = D_xxe(4, 2) = Div(- 8.0, 45.0);
  D_xxe(2, 5) = D_xxe(5, 2) = Div( 16.0, 45.0);
  D_xxe(2, 6) = D_xxe(6, 2) = Div(- 8.0, 45.0);
  D_xxe(2, 7) = D_xxe(7, 2) = Div(  4.0, 45.0);
  D_xxe(2, 8) = D_xxe(8, 2) = Div(- 8.0, 45.0);
  D_xxe(2, 9) = D_xxe(9, 2) = Div(  4.0, 45.0);

  D_xxe(3, 3)               = Div( 14.0, 45.0);
  D_xxe(3, 4) = D_xxe(4, 3) = Div(  1.0, 45.0);
  D_xxe(3, 5) = D_xxe(5, 3) = Div(- 8.0, 45.0);
  D_xxe(3, 6) = D_xxe(6, 3) = Div(  7.0, 45.0);
  D_xxe(3, 7) = D_xxe(7, 3) = Div(- 1.0, 90.0);
  D_xxe(3, 8) = D_xxe(8, 3) = Div(  4.0, 45.0);
  D_xxe(3, 9) = D_xxe(9, 3) = Div(- 7.0, 90.0);

  D_xxe(4, 4)               = Div( 56.0, 45.0);
  D_xxe(4, 5) = D_xxe(5, 4) = Div(-64.0, 45.0);
  D_xxe(4, 6) = D_xxe(6, 4) = Div(  8.0, 45.0);
  D_xxe(4, 7) = D_xxe(7, 4) = Div(  7.0, 45.0);
  D_xxe(4, 8) = D_xxe(8, 4) = Div(- 8.0, 45.0);
  D_xxe(4, 9) = D_xxe(9, 4) = Div(  1.0, 45.0);

  D_xxe(5, 5)               = Div(128.0, 45.0);
  D_xxe(5, 6) = D_xxe(6, 5) = Div(-64.0, 45.0);
  D_xxe(5, 7) = D_xxe(7, 5) = Div(- 8.0, 45.0);
  D_xxe(5, 8) = D_xxe(8, 5) = Div( 16.0, 45.0);
  D_xxe(5, 9) = D_xxe(9, 5) = Div(- 8.0, 45.0);

  D_xxe(6, 6)               = Div( 56.0, 45.0);
  D_xxe(6, 7) = D_xxe(7, 6) = Div(  1.0, 45.0);
  D_xxe(6, 8) = D_xxe(8, 6) = Div(- 8.0, 45.0);
  D_xxe(6, 9) = D_xxe(9, 6) = Div(  7.0, 45.0);

  D_xxe(7, 7)               = Div( 14.0, 45.0);
  D_xxe(7, 8) = D_xxe(8, 7) = Div(-16.0, 45.0);
  D_xxe(7, 9) = D_xxe(9, 7) = Div(  2.0, 45.0);

  D_xxe(8, 8)               = Div( 32.0, 45.0);
  D_xxe(8, 9) = D_xxe(9, 8) = Div(- 16.0, 45.0);

  D_xxe(9, 9)               = Div( 14.0, 45.0);


/*  (∂_yφ , ∂_yφ) の計算  */
  D_yye(1, 1)               = Div( 14.0, 45.0);
  D_yye(1, 2) = D_yye(2, 1) = Div(  7.0, 45.0);
  D_yye(1, 3) = D_yye(3, 1) = Div(- 7.0, 90.0);
  D_yye(1, 4) = D_yye(4, 1) = Div(-16.0, 45.0);
  D_yye(1, 5) = D_yye(5, 1) = Div(- 8.0, 45.0);
  D_yye(1, 6) = D_yye(6, 1) = Div(  4.0, 45.0);
  D_yye(1, 7) = D_yye(7, 1) = Div(  2.0, 45.0);
  D_yye(1, 8) = D_yye(8, 1) = Div(  1.0, 45.0);
  D_yye(1, 9) = D_yye(9, 1) = Div(- 1.0, 90.0);

  D_yye(2, 2)               = Div( 56.0, 45.0);
  D_yye(2, 3) = D_yye(3, 2) = Div(  7.0, 45.0);
  D_yye(2, 4) = D_yye(4, 2) = Div(- 8.0, 45.0);
  D_yye(2, 5) = D_yye(5, 2) = Div(-64.0, 45.0);
  D_yye(2, 6) = D_yye(6, 2) = Div(- 8.0, 45.0);
```

```
  D_yye(2, 7) = D_yye(7, 2) = Div(  1.0, 45.0);
  D_yye(2, 8) = D_yye(8, 2) = Div(  8.0, 45.0);
  D_yye(2, 9) = D_yye(9, 2) = Div(  1.0, 45.0);

  D_yye(3, 3)               = Div( 14.0, 45.0);
  D_yye(3, 4) = D_yye(4, 3) = Div(  4.0, 45.0);
  D_yye(3, 5) = D_yye(5, 3) = Div(- 8.0, 45.0);
  D_yye(3, 6) = D_yye(6, 3) = Div(-16.0, 45.0);
  D_yye(3, 7) = D_yye(7, 3) = Div(- 1.0, 90.0);
  D_yye(3, 8) = D_yye(8, 3) = Div(  1.0, 45.0);
  D_yye(3, 9) = D_yye(9, 3) = Div(  2.0, 45.0);

  D_yye(4, 4)               = Div( 32.0, 45.0);
  D_yye(4, 5) = D_yye(5, 4) = Div( 16.0, 45.0);
  D_yye(4, 6) = D_yye(6, 4) = Div(- 8.0, 45.0);
  D_yye(4, 7) = D_yye(7, 4) = Div(-16.0, 45.0);
  D_yye(4, 8) = D_yye(8, 4) = Div(- 8.0, 45.0);
  D_yye(4, 9) = D_yye(9, 4) = Div(  4.0, 45.0);

  D_yye(5, 5)               = Div(128.0, 45.0);
  D_yye(5, 6) = D_yye(6, 5) = Div( 16.0, 45.0);
  D_yye(5, 7) = D_yye(7, 5) = Div(- 8.0, 45.0);
  D_yye(5, 8) = D_yye(8, 5) = Div(-64.0, 45.0);
  D_yye(5, 9) = D_yye(9, 5) = Div(- 8.0, 45.0);

  D_yye(6, 6)               = Div( 32.0, 45.0);
  D_yye(6, 7) = D_yye(7, 6) = Div(  4.0, 45.0);
  D_yye(6, 8) = D_yye(8, 6) = Div(- 8.0, 45.0);
  D_yye(6, 9) = D_yye(9, 6) = Div(-16.0, 45.0);

  D_yye(7, 7)               = Div( 14.0, 45.0);
  D_yye(7, 8) = D_yye(8, 7) = Div(  7.0, 45.0);
  D_yye(7, 9) = D_yye(9, 7) = Div(- 7.0, 90.0);

  D_yye(8, 8)               = Div( 56.0, 45.0);
  D_yye(8, 9) = D_yye(9, 8) = Div(  7.0, 45.0);

  D_yye(9, 9)               = Div( 14.0, 45.0);


/*  (∂_yφ, ∂_xφ) の計算  */
  D_yxe(1, 1) = Div(  1.0,  4.0);
  D_yxe(1, 2) = Div(- 1.0,  3.0);
  D_yxe(1, 3) = Div(  1.0, 12.0);
  D_yxe(1, 4) = Div(  1.0,  3.0);
  D_yxe(1, 5) = Div(- 4.0,  9.0);
  D_yxe(1, 6) = Div(  1.0,  9.0);
  D_yxe(1, 7) = Div(- 1.0, 12.0);
  D_yxe(1, 8) = Div(  1.0,  9.0);
  D_yxe(1, 9) = Div(- 1.0, 36.0);

  D_yxe(2, 1) = Div(  1.0,  3.0);
  D_yxe(2, 2) = 0.0;
  D_yxe(2, 3) = Div(- 1.0,  3.0);
  D_yxe(2, 4) = Div(  4.0,  9.0);
  D_yxe(2, 5) = 0.0;
  D_yxe(2, 6) = Div(- 4.0,  9.0);
  D_yxe(2, 7) = Div(- 1.0,  9.0);
  D_yxe(2, 8) = 0.0;
  D_yxe(2, 9) = Div(  1.0, 9.0);

  D_yxe(3, 1) = Div(- 1.0, 12.0);
  D_yxe(3, 2) = Div(  1.0,  3.0);
  D_yxe(3, 3) = Div(- 1.0,  4.0);
  D_yxe(3, 4) = Div(- 1.0,  9.0);
  D_yxe(3, 5) = Div(  4.0,  9.0);
  D_yxe(3, 6) = Div(- 1.0,  3.0);
  D_yxe(3, 7) = Div(  1.0, 36.0);
  D_yxe(3, 8) = Div(- 1.0,  9.0);
  D_yxe(3, 9) = Div(  1.0, 12.0);

  D_yxe(4, 1) = Div(- 1.0,  3.0);
  D_yxe(4, 2) = Div(  4.0,  9.0);
```

```
  D_yxe(4, 3) = Div(- 1.0,  9.0);
  D_yxe(4, 4) = 0.0;
  D_yxe(4, 5) = 0.0;
  D_yxe(4, 6) = 0.0;
  D_yxe(4, 7) = Div(  1.0,  3.0);
  D_yxe(4, 8) = Div(- 4.0,  9.0);
  D_yxe(4, 9) = Div(  1.0,  9.0);

  D_yxe(5, 1) = Div(- 4.0,  9.0);
  D_yxe(5, 2) = 0.0;
  D_yxe(5, 3) = Div(  4.0,  9.0);
  D_yxe(5, 4) = 0.0;
  D_yxe(5, 5) = 0.0;
  D_yxe(5, 6) = 0.0;
  D_yxe(5, 7) = Div(  4.0,  9.0);
  D_yxe(5, 8) = 0.0;
  D_yxe(5, 9) = Div(- 4.0,  9.0);

  D_yxe(6, 1) = Div(  1.0,  9.0);
  D_yxe(6, 2) = Div(- 4.0,  9.0);
  D_yxe(6, 3) = Div(  1.0,  3.0);
  D_yxe(6, 4) = 0.0;
  D_yxe(6, 5) = 0.0;
  D_yxe(6, 6) = 0.0;
  D_yxe(6, 7) = Div(- 1.0,  9.0);
  D_yxe(6, 8) = Div(  4.0,  9.0);
  D_yxe(6, 9) = Div(- 1.0,  3.0);

  D_yxe(7, 1) = Div(  1.0, 12.0);
  D_yxe(7, 2) = Div(- 1.0,  9.0);
  D_yxe(7, 3) = Div(  1.0, 36.0);
  D_yxe(7, 4) = Div(- 1.0,  3.0);
  D_yxe(7, 5) = Div(  4.0,  9.0);
  D_yxe(7, 6) = Div(- 1.0,  9.0);
  D_yxe(7, 7) = Div(- 1.0,  4.0);
  D_yxe(7, 8) = Div(  1.0,  3.0);
  D_yxe(7, 9) = Div(- 1.0, 12.0);

  D_yxe(8, 1) = Div(  1.0,  9.0);
  D_yxe(8, 2) = 0.0;
  D_yxe(8, 3) = Div(- 1.0,  9.0);
  D_yxe(8, 4) = Div(- 4.0,  9.0);
  D_yxe(8, 5) = 0.0;
  D_yxe(8, 6) = Div(  4.0,  9.0);
  D_yxe(8, 7) = Div(- 1.0,  3.0);
  D_yxe(8, 8) = 0.0;
  D_yxe(8, 9) = Div(  1.0,  3.0);

  D_yxe(9, 1) = Div(- 1.0, 36.0);
  D_yxe(9, 2) = Div(  1.0, 9.0);
  D_yxe(9, 3) = Div(- 1.0, 12.0);
  D_yxe(9, 4) = Div(  1.0,  9.0);
  D_yxe(9, 5) = Div(- 4.0,  9.0);
  D_yxe(9, 6) = Div(  1.0,  3.0);
  D_yxe(9, 7) = Div(  1.0, 12.0);
  D_yxe(9, 8) = Div(- 1.0,  3.0);
  D_yxe(9, 9) = Div(  1.0,  4.0);


/*  (φφ, φ) の計算  */
  Ae[0][0][0]                                 = Div(  9.0, 19600.0) * h2;
  Ae[0][0][1] = Ae[0][1][0] = Ae[1][0][0] = Div(  1.0,   980.0) * h2;
  Ae[0][0][2] = Ae[0][2][0] = Ae[2][0][0] = Div(- 3.0, 19600.0) * h2;
  Ae[0][0][3] = Ae[0][3][0] = Ae[3][0][0] = Div(  1.0,   980.0) * h2;
  Ae[0][0][4] = Ae[0][4][0] = Ae[4][0][0] = Div(  1.0,   441.0) * h2;
  Ae[0][0][5] = Ae[0][5][0] = Ae[5][0][0] = Div(- 1.0,  2940.0) * h2;
  Ae[0][0][6] = Ae[0][6][0] = Ae[6][0][0] = Div(- 3.0, 19600.0) * h2;
  Ae[0][0][7] = Ae[0][7][0] = Ae[7][0][0] = Div(- 1.0,  2940.0) * h2;
  Ae[0][0][8] = Ae[0][8][0] = Ae[8][0][0] = Div(  1.0, 19600.0) * h2;

  Ae[0][1][1] = Ae[1][0][1] = Ae[1][1][0]
                  = Div(  1.0,  1225.0) * h2;
  Ae[0][1][2] = Ae[0][2][1] = Ae[1][0][2] = Ae[1][2][0] = Ae[2][0][1]
```

```
  = Ae[2][1][0] = Div(- 1.0,  2450.0) * h2;
Ae[0][1][3] = Ae[0][3][1] = Ae[1][0][3] = Ae[1][3][0] = Ae[3][0][1]
  = Ae[3][1][0] = Div(  1.0,   441.0) * h2;
Ae[0][1][4] = Ae[0][4][1] = Ae[1][0][4] = Ae[1][4][0] = Ae[4][0][1]
  = Ae[4][1][0] = Div(  4.0,  2205.0) * h2;
Ae[0][1][5] = Ae[0][5][1] = Ae[1][0][5] = Ae[1][5][0] = Ae[5][0][1]
  = Ae[5][1][0] = Div(- 2.0,  2205.0) * h2;
Ae[0][1][6] = Ae[0][6][1] = Ae[1][0][6] = Ae[1][6][0] = Ae[6][0][1]
  = Ae[6][1][0] = Div(- 1.0,  2940.0) * h2;
Ae[0][1][7] = Ae[0][7][1] = Ae[1][0][7] = Ae[1][7][0] = Ae[7][0][1]
  = Ae[7][1][0] = Div(- 1.0,  3675.0) * h2;
Ae[0][1][8] = Ae[0][8][1] = Ae[1][0][8] = Ae[1][8][0] = Ae[8][0][1]
  = Ae[8][1][0] = Div(  1.0,  7350.0) * h2;

Ae[0][2][2] = Ae[2][0][2] = Ae[2][0][0] = Div(- 3.0, 19600.0) * h2;
Ae[0][2][3] = Ae[0][3][2] = Ae[2][0][3] = Ae[2][3][0] = Ae[3][0][2]
  = Ae[3][2][0] = Div(- 1.0,  2940.0) * h2;
Ae[0][2][4] = Ae[0][4][2] = Ae[2][0][4] = Ae[2][4][0] = Ae[4][0][2]
  = Ae[4][2][0] = Div(- 2.0,  2205.0) * h2;
Ae[0][2][5] = Ae[0][5][2] = Ae[2][0][5] = Ae[2][5][0] = Ae[5][0][2]
  = Ae[5][2][0] = Div(- 1.0,  2940.0) * h2;
Ae[0][2][6] = Ae[0][6][2] = Ae[2][0][6] = Ae[2][6][0] = Ae[6][0][2]
  = Ae[6][2][0] = Div(  1.0, 19600.0) * h2;
Ae[0][2][7] = Ae[0][7][2] = Ae[2][0][7] = Ae[2][7][0] = Ae[7][0][2]
  = Ae[7][2][0] = Div(  1.0,  7350.0) * h2;
Ae[0][2][8] = Ae[0][8][2] = Ae[2][0][8] = Ae[2][8][0] = Ae[8][0][2]
  = Ae[8][2][0] = Div(  1.0, 19600.0) * h2;

Ae[0][3][3] = Ae[3][0][3] = Ae[3][3][0] = Div(  1.0,  1225.0) * h2;
Ae[0][3][4] = Ae[0][4][3] = Ae[3][0][4] = Ae[3][4][0] = Ae[4][0][3]
  = Ae[4][3][0] = Div(  4.0,  2205.0) * h2;
Ae[0][3][5] = Ae[0][5][3] = Ae[3][0][5] = Ae[3][5][0] = Ae[5][0][3]
  = Ae[5][3][0] = Div(- 1.0,  3675.0) * h2;
Ae[0][3][6] = Ae[0][6][3] = Ae[3][0][6] = Ae[3][6][0] = Ae[6][0][3]
  = Ae[6][3][0] = Div(- 1.0,  2450.0) * h2;
Ae[0][3][7] = Ae[0][7][3] = Ae[3][0][7] = Ae[3][7][0] = Ae[7][0][3]
  = Ae[7][3][0] = Div(- 2.0,  2205.0) * h2;
Ae[0][3][8] = Ae[0][8][3] = Ae[3][0][8] = Ae[3][8][0] = Ae[8][0][3]
  = Ae[8][3][0] = Div(  1.0,  7350.0) * h2;

Ae[0][4][4] = Ae[4][0][4] = Ae[4][4][0] = Div( 16.0, 11025.0) * h2;
Ae[0][4][5] = Ae[0][5][4] = Ae[4][0][5] = Ae[4][5][0] = Ae[5][0][4]
  = Ae[5][4][0] = Div(- 8.0, 11025.0) * h2;
Ae[0][4][6] = Ae[0][6][4] = Ae[4][0][6] = Ae[4][6][0] = Ae[6][0][4]
  = Ae[6][4][0] = Div(- 2.0,  2205.0) * h2;
Ae[0][4][7] = Ae[0][7][4] = Ae[4][0][7] = Ae[4][7][0] = Ae[7][0][4]
  = Ae[7][4][0] = Div(- 8.0, 11025.0) * h2;
Ae[0][4][8] = Ae[0][8][4] = Ae[4][0][8] = Ae[4][8][0] = Ae[8][0][4]
  = Ae[8][4][0] = Div(  4.0, 11025.0) * h2;

Ae[0][5][5] = Ae[5][0][5] = Ae[5][5][0] = Div(- 1.0,  3675.0) * h2;
Ae[0][5][6] = Ae[0][6][5] = Ae[5][0][6] = Ae[5][6][0] = Ae[6][0][5]
  = Ae[6][5][0] = Div(  1.0,  7350.0) * h2;
Ae[0][5][7] = Ae[0][7][5] = Ae[5][0][7] = Ae[5][7][0] = Ae[7][0][5]
  = Ae[7][5][0] = Div(  4.0, 11025.0) * h2;
Ae[0][5][8] = Ae[0][8][5] = Ae[5][0][8] = Ae[5][8][0] = Ae[8][0][5]
  = Ae[8][5][0] = Div(  1.0,  7350.0) * h2;

Ae[0][6][6] = Ae[6][0][6] = Ae[6][6][0] = Div(- 3.0, 19600.0) * h2;
Ae[0][6][7] = Ae[0][7][6] = Ae[6][0][7] = Ae[6][7][0] = Ae[7][0][6]
  = Ae[7][6][0] = Div(- 1.0,  2940.0) * h2;
Ae[0][6][8] = Ae[0][8][6] = Ae[6][0][8] = Ae[6][8][0] = Ae[8][0][6]
  = Ae[8][6][0] = Div(  1.0, 19600.0) * h2;

Ae[0][7][7] = Ae[7][0][7] = Ae[7][7][0] = Div(- 1.0,  3675.0) * h2;
Ae[0][7][8] = Ae[0][8][7] = Ae[7][0][8] = Ae[7][8][0] = Ae[8][0][7]
  = Ae[8][7][0] = Div(  1.0,  7350.0) * h2;

Ae[0][8][8] = Ae[8][0][8] = Ae[8][8][0] = Div(  1.0, 19600.0) * h2;


Ae[1][1][1] = Div( 12.0,  1225.0) * h2;
Ae[1][1][2] = Ae[1][2][1] = Ae[2][1][1] = Div(  1.0,  1225.0) * h2;
```

```
Ae[1][1][3] = Ae[1][3][1] = Ae[3][1][1] = Div(  4.0,  2205.0) * h2;
Ae[1][1][4] = Ae[1][4][1] = Ae[4][1][1] = Div( 16.0,   735.0) * h2;
Ae[1][1][5] = Ae[1][5][1] = Ae[5][1][1] = Div(  4.0,  2205.0) * h2;
Ae[1][1][6] = Ae[1][6][1] = Ae[6][1][1] = Div(- 1.0,  3675.0) * h2;
Ae[1][1][7] = Ae[1][7][1] = Ae[7][1][1] = Div(- 4.0,  1225.0) * h2;
Ae[1][1][8] = Ae[1][8][1] = Ae[8][1][1] = Div(- 1.0,  3675.0) * h2;

Ae[1][2][2] = Ae[2][1][2] = Ae[2][2][1] = Div(  1.0,   980.0) * h2;
Ae[1][2][3] = Ae[1][3][2] = Ae[2][1][3] = Ae[2][3][1] = Ae[3][1][2]
  = Ae[3][2][1] = Div(- 2.0,  2205.0) * h2;
Ae[1][2][4] = Ae[1][4][2] = Ae[2][1][4] = Ae[2][4][1] = Ae[4][1][2]
  = Ae[4][2][1] = Div(  4.0,  2205.0) * h2;
Ae[1][2][5] = Ae[1][5][2] = Ae[2][1][5] = Ae[2][5][1] = Ae[5][1][2]
  = Ae[5][2][1] = Div(  1.0,   441.0) * h2;
Ae[1][2][6] = Ae[1][6][2] = Ae[2][1][6] = Ae[2][6][1] = Ae[6][1][2]
  = Ae[6][2][1] = Div(  1.0,  7350.0) * h2;
Ae[1][2][7] = Ae[1][7][2] = Ae[2][1][7] = Ae[2][7][1] = Ae[7][1][2]
  = Ae[7][2][1] = Div(- 1.0,  3675.0) * h2;
Ae[1][2][8] = Ae[1][8][2] = Ae[2][1][8] = Ae[2][8][1] = Ae[8][1][2]
  = Ae[8][2][1] = Div(- 1.0,  2940.0) * h2;


Ae[1][3][3] = Ae[3][1][3] = Ae[3][3][1] = Div(  4.0,  2205.0) * h2;
Ae[1][3][4] = Ae[1][4][3] = Ae[3][1][4] = Ae[3][4][1] = Ae[4][1][3]
  = Ae[4][3][1] = Div( 16.0, 11025.0) * h2;
Ae[1][3][5] = Ae[1][5][3] = Ae[3][1][5] = Ae[3][5][1] = Ae[5][1][3]
  = Ae[5][3][1] = Div(- 8.0, 11025.0) * h2;
Ae[1][3][6] = Ae[1][6][3] = Ae[3][1][6] = Ae[3][6][1] = Ae[6][1][3]
  = Ae[6][3][1] = Div(- 2.0,  2205.0) * h2;
Ae[1][3][7] = Ae[1][7][3] = Ae[3][1][7] = Ae[3][7][1] = Ae[7][1][3]
  = Ae[7][3][1] = Div(- 8.0, 11025.0) * h2;
Ae[1][3][8] = Ae[1][8][3] = Ae[3][1][8] = Ae[3][8][1] = Ae[8][1][3]
  = Ae[8][3][1] = Div(  4.0, 11025.0) * h2;

Ae[1][4][4] = Ae[4][1][4] = Ae[4][4][1] = Div( 64.0,  3675.0) * h2;
Ae[1][4][5] = Ae[1][5][4] = Ae[4][1][5] = Ae[4][5][1] = Ae[5][1][4]
  = Ae[5][4][1] = Div( 16.0, 11025.0) * h2;
Ae[1][4][6] = Ae[1][6][4] = Ae[4][1][6] = Ae[4][6][1] = Ae[6][1][4]
  = Ae[6][4][1] = Div(- 8.0, 11025.0) * h2;
Ae[1][4][7] = Ae[1][7][4] = Ae[4][1][7] = Ae[4][7][1] = Ae[7][1][4]
  = Ae[7][4][1] = Div(-32.0,  3675.0) * h2;
Ae[1][4][8] = Ae[1][8][4] = Ae[4][1][8] = Ae[4][8][1] = Ae[8][1][4]
  = Ae[8][4][1] = Div(- 8.0, 11025.0) * h2;

Ae[1][5][5] = Ae[5][1][5] = Ae[5][5][1] = Div(  4.0,  2205.0) * h2;
Ae[1][5][6] = Ae[1][6][5] = Ae[5][1][6] = Ae[5][6][1] = Ae[6][1][5]
  = Ae[6][5][1] = Div(  4.0, 11025.0) * h2;
Ae[1][5][7] = Ae[1][7][5] = Ae[5][1][7] = Ae[5][7][1] = Ae[7][1][5]
  = Ae[7][5][1] = Div(- 8.0, 11025.0) * h2;
Ae[1][5][8] = Ae[1][8][5] = Ae[5][1][8] = Ae[5][8][1] = Ae[8][1][5]
  = Ae[8][5][1] = Div(- 2.0,  2205.0) * h2;

Ae[1][6][6] = Ae[6][1][6] = Ae[6][6][1] = Div(- 1.0,  2940.0) * h2;
Ae[1][6][7] = Ae[1][7][6] = Ae[6][1][7] = Ae[6][7][1] = Ae[7][1][6]
  = Ae[7][6][1] = Div(- 1.0,  3675.0) * h2;
Ae[1][6][8] = Ae[1][8][6] = Ae[6][1][8] = Ae[6][8][1] = Ae[8][1][6]
  = Ae[8][6][1] = Div(  1.0,  7350.0) * h2;

Ae[1][7][7] = Ae[7][1][7] = Ae[7][7][1] = Div(- 4.0,  1225.0) * h2;
Ae[1][7][8] = Ae[1][8][7] = Ae[7][1][8] = Ae[7][8][1] = Ae[8][1][7]
  = Ae[8][7][1] = Div(- 1.0,  3675.0) * h2;

Ae[1][8][8] = Ae[8][1][8] = Ae[8][8][1] = Div(- 1.0,  2940.0) * h2;

Ae[2][2][2] = Div(  9.0, 19600.0) * h2;
Ae[2][2][3] = Ae[2][3][2] = Ae[3][2][2] = Div(- 1.0,  2940.0) * h2;
Ae[2][2][4] = Ae[2][4][2] = Ae[4][2][2] = Div(  1.0,   441.0) * h2;
Ae[2][2][5] = Ae[2][5][2] = Ae[5][2][2] = Div(  1.0,   980.0) * h2;
Ae[2][2][6] = Ae[2][6][2] = Ae[6][2][2] = Div(  1.0, 19600.0) * h2;
Ae[2][2][7] = Ae[2][7][2] = Ae[7][2][2] = Div(- 1.0,  2940.0) * h2;
Ae[2][2][8] = Ae[2][8][2] = Ae[8][2][2] = Div(- 3.0, 19600.0) * h2;

Ae[2][3][3] = Ae[3][2][3] = Ae[3][3][2] = Div(- 1.0,  3675.0) * h2;
```

```
Ae[2][3][4] = Ae[2][4][3] = Ae[3][2][4] = Ae[3][4][2] = Ae[4][2][3]
  = Ae[4][3][2] = Div(- 8.0, 11025.0) * h2;
Ae[2][3][5] = Ae[2][5][3] = Ae[3][2][5] = Ae[3][5][2] = Ae[5][2][3]
  = Ae[5][3][2] = Div(- 1.0,  3675.0) * h2;
Ae[2][3][6] = Ae[2][6][3] = Ae[3][2][6] = Ae[3][6][2] = Ae[6][2][3]
  = Ae[6][3][2] = Div(  1.0,  7350.0) * h2;
Ae[2][3][7] = Ae[2][7][3] = Ae[3][2][7] = Ae[3][7][2] = Ae[7][2][3]
  = Ae[7][3][2] = Div(  4.0, 11025.0) * h2;
Ae[2][3][8] = Ae[2][8][3] = Ae[3][2][8] = Ae[3][8][2] = Ae[8][2][3]
  = Ae[8][3][2] = Div(  1.0,  7350.0) * h2;

Ae[2][4][4] = Ae[4][2][4] = Ae[4][4][2] = Div( 16.0, 11025.0) * h2;
Ae[2][4][5] = Ae[2][5][4] = Ae[4][2][5] = Ae[4][5][2] = Ae[5][2][4]
  = Ae[5][4][2] = Div(  4.0,  2205.0) * h2;
Ae[2][4][6] = Ae[2][6][4] = Ae[4][2][6] = Ae[4][6][2] = Ae[6][2][4]
  = Ae[6][4][2] = Div(  4.0, 11025.0) * h2;
Ae[2][4][7] = Ae[2][7][4] = Ae[4][2][7] = Ae[4][7][2] = Ae[7][2][4]
  = Ae[7][4][2] = Div(- 8.0, 11025.0) * h2;
Ae[2][4][8] = Ae[2][8][4] = Ae[4][2][8] = Ae[4][8][2] = Ae[8][2][4]
  = Ae[8][4][2] = Div(- 2.0,  2205.0) * h2;

Ae[2][5][5] = Ae[5][2][5] = Ae[5][5][2] = Div(  1.0,   980.0) * h2;
Ae[2][5][6] = Ae[2][6][5] = Ae[5][2][6] = Ae[5][6][2] = Ae[6][2][5]
  = Ae[6][5][2] = Div(  1.0,  7350.0) * h2;
Ae[2][5][7] = Ae[2][7][5] = Ae[5][2][7] = Ae[5][7][2] = Ae[7][2][5]
  = Ae[7][5][2] = Div(- 2.0,  2205.0) * h2;
Ae[2][5][8] = Ae[2][8][5] = Ae[5][2][8] = Ae[5][8][2] = Ae[8][2][5]
  = Ae[8][5][2] = Div(- 1.0,  2450.0) * h2;

Ae[2][6][6] = Ae[6][2][6] = Ae[6][6][2] = Div(  1.0, 19600.0) * h2;
Ae[2][6][7] = Ae[2][7][6] = Ae[6][2][7] = Ae[6][7][2] = Ae[7][2][6]
  = Ae[7][6][2] = Div(  1.0,  7350.0) * h2;
Ae[2][6][8] = Ae[2][8][6] = Ae[6][2][8] = Ae[6][8][2] = Ae[8][2][6]
  = Ae[8][6][2] = Div(  1.0, 19600.0) * h2;

Ae[2][7][7] = Ae[7][2][7] = Ae[7][7][2] = Div(- 1.0,  3675.0) * h2;
Ae[2][7][8] = Ae[2][8][7] = Ae[7][2][8] = Ae[7][8][2] = Ae[8][2][7]
  = Ae[8][7][2] = Div(- 1.0,  2940.0) * h2;

Ae[2][8][8] = Ae[8][2][8] = Ae[8][8][2] = Div(- 3.0, 19600.0) * h2;


Ae[3][3][3] = Div( 12.0,  1225.0) * h2;
Ae[3][3][4] = Ae[3][4][3] = Ae[4][3][3] =Div( 16.0,   735.0) * h2;
Ae[3][3][5] = Ae[3][5][3] = Ae[5][3][3] =Div(- 4.0,  1225.0) * h2;
Ae[3][3][6] = Ae[3][6][3] = Ae[6][3][3] =Div(  1.0,  1225.0) * h2;
Ae[3][3][7] = Ae[3][7][3] = Ae[7][3][3] =Div(  4.0,  2205.0) * h2;
Ae[3][3][8] = Ae[3][8][3] = Ae[8][3][3] =Div(- 1.0,  3675.0) * h2;

Ae[3][4][4] = Ae[4][3][4] = Ae[4][4][3] = Div( 64.0,  3675.0) * h2;
Ae[3][4][5] = Ae[3][5][4] = Ae[4][3][5] = Ae[4][5][3] = Ae[5][3][4]
= Ae[5][4][3] =Div(-32.0,  3675.0) * h2;
Ae[3][4][6] = Ae[3][6][4] = Ae[4][3][6] = Ae[4][6][3] = Ae[6][3][4]
= Ae[6][4][3] = Div(  4.0,  2205.0) * h2;
Ae[3][4][7] = Ae[3][7][4] = Ae[4][3][7] = Ae[4][7][3] = Ae[7][3][4]
= Ae[7][4][3] = Div( 16.0, 11025.0) * h2;
Ae[3][4][8] = Ae[3][8][4] = Ae[4][3][8] = Ae[4][8][3] = Ae[8][3][4]
= Ae[8][4][3] = Div(- 8.0, 11025.0) * h2;

Ae[3][5][5] = Ae[5][3][5] = Ae[5][5][3] = Div(- 4.0,  1225.0) * h2;
Ae[3][5][6] = Ae[3][6][5] = Ae[5][3][6] = Ae[5][6][3] = Ae[6][3][5]
= Ae[6][5][3] = Div(- 1.0,  3675.0) * h2;
Ae[3][5][7] = Ae[3][7][5] = Ae[5][3][7] = Ae[5][7][3] = Ae[7][3][5]
= Ae[7][5][3] = Div(- 8.0, 11025.0) * h2;
Ae[3][5][8] = Ae[3][8][5] = Ae[5][3][8] = Ae[5][8][3] = Ae[8][3][5]
= Ae[8][5][3] = Div(- 1.0,  3675.0) * h2;

Ae[3][6][6] = Ae[6][3][6] = Ae[6][6][3] = Div(  1.0,   980.0) * h2;
Ae[3][6][7] = Ae[3][7][6] = Ae[6][3][7] = Ae[6][7][3] = Ae[7][3][6]
= Ae[7][6][3] = Div(  1.0,   441.0) * h2;
Ae[3][6][8] = Ae[3][8][6] = Ae[6][3][8] = Ae[6][8][3] = Ae[8][3][6]
= Ae[8][6][3] = Div(- 1.0,  2940.0) * h2;
```

```
  Ae[3][7][7] = Ae[7][3][7] = Ae[7][7][3] = Div(  4.0,  2205.0) * h2;
  Ae[3][7][8] = Ae[3][8][7] = Ae[7][3][8] = Ae[7][8][3] = Ae[8][3][7]
  = Ae[8][7][3] = Div(- 2.0,  2205.0) * h2;

  Ae[3][8][8] = Ae[8][3][8] = Ae[8][8][3] = Div(- 1.0,  2940.0) * h2;


  Ae[4][4][4] = Div(256.0,  1225.0) * h2;
  Ae[4][4][5] = Ae[4][5][4] = Ae[5][4][4] = Div( 64.0,  3675.0) * h2;
  Ae[4][4][6] = Ae[4][6][4] = Ae[6][4][4] = Div( 16.0, 11025.0) * h2;
  Ae[4][4][7] = Ae[4][7][4] = Ae[7][4][4] = Div( 64.0,  3675.0) * h2;
  Ae[4][4][8] = Ae[4][8][4] = Ae[8][4][4] = Div( 16.0, 11025.0) * h2;

  Ae[4][5][5] = Ae[5][4][5] = Ae[5][5][4] = Div( 16.0,   735.0) * h2;
  Ae[4][5][6] = Ae[4][6][5] = Ae[5][4][6] = Ae[5][6][4] = Ae[6][4][5]
  = Ae[6][5][4] = Div(- 8.0, 11025.0) * h2;
  Ae[4][5][7] = Ae[4][7][5] = Ae[5][4][7] = Ae[5][7][4] = Ae[7][4][5]
  = Ae[7][5][4] = Div( 16.0, 11025.0) * h2;
  Ae[4][5][8] = Ae[4][8][5] = Ae[5][4][8] = Ae[5][8][4] = Ae[8][4][5]
  = Ae[8][5][4] = Div(  4.0,  2205.0) * h2;

  Ae[4][6][6] = Ae[6][4][6] = Ae[6][6][4] = Div(  1.0,   441.0) * h2;
  Ae[4][6][7] = Ae[4][7][6] = Ae[6][4][7] = Ae[6][7][4] = Ae[7][4][6]
  = Ae[7][6][4] = Div(  4.0,  2205.0) * h2;
  Ae[4][6][8] = Ae[4][8][6] = Ae[6][4][8] = Ae[6][8][4] = Ae[8][4][6]
  = Ae[8][6][4] = Div(- 2.0,  2205.0) * h2;

  Ae[4][7][7] = Ae[7][4][7] = Ae[7][7][4] = Div( 16.0,   735.0) * h2;
  Ae[4][7][8] = Ae[4][8][7] = Ae[7][4][8] = Ae[7][8][4] = Ae[8][4][7]
  = Ae[8][7][4] = Div(  4.0,  2205.0) * h2;

  Ae[4][8][8] = Ae[8][4][8] = Ae[8][8][4] = Div(  1.0,   441.0) * h2;


  Ae[5][5][5] = Div( 12.0,  1225.0) * h2;
  Ae[5][5][6] = Ae[5][6][5] = Ae[6][5][5] = Div(- 1.0,  3675.0) * h2;
  Ae[5][5][7] = Ae[5][7][5] = Ae[7][5][5] = Div(  4.0,  2205.0) * h2;
  Ae[5][5][8] = Ae[5][8][5] = Ae[8][5][5] = Div(  1.0,  1225.0) * h2;

  Ae[5][6][6] = Ae[6][5][6] = Ae[6][6][5] = Div(- 1.0,  2940.0) * h2;
  Ae[5][6][7] = Ae[5][7][6] = Ae[6][5][7] = Ae[6][7][5] = Ae[7][5][6]
  = Ae[7][6][5] = Div(- 2.0,  2205.0) * h2;
  Ae[5][6][8] = Ae[5][8][6] = Ae[6][5][8] = Ae[6][8][5] = Ae[8][5][6]
  = Ae[8][6][5] = Div(- 1.0,  2940.0) * h2;

  Ae[5][7][7] = Ae[7][5][7] = Ae[7][7][5] = Div(  4.0,  2205.0) * h2;
  Ae[5][7][8] = Ae[5][8][7] = Ae[7][5][8] = Ae[7][8][5] = Ae[8][5][7]
  = Ae[8][7][5] = Div(  1.0,   441.0) * h2;

  Ae[5][8][8] = Ae[8][5][8] = Ae[8][8][5] = Div(  1.0,   980.0) * h2;


  Ae[6][6][6] = Div(  9.0, 19600.0) * h2;
  Ae[6][6][7] = Ae[6][7][6] = Ae[7][6][6] = Div(  1.0,   980.0) * h2;
  Ae[6][6][8] = Ae[6][8][6] = Ae[8][6][6] = Div(- 3.0, 19600.0) * h2;

  Ae[6][7][7] = Ae[7][6][7] = Ae[7][7][6] = Div(  1.0,  1225.0) * h2;
  Ae[6][7][8] = Ae[6][8][7] = Ae[7][6][8] = Ae[7][8][6] = Ae[8][6][7]
  = Ae[8][7][6] = Div( -1.0,  2450.0) * h2;

  Ae[6][8][8] = Ae[8][6][8] = Ae[8][8][6] = Div(- 3.0, 19600.0) * h2;


  Ae[7][7][7] = Div( 12.0,  1225.0) * h2;
  Ae[7][7][8] = Ae[7][8][7] = Ae[8][7][7] = Div(  1.0,  1225.0) * h2;

  Ae[7][8][8] = Ae[8][7][8] = Ae[8][8][7] = Div(  1.0,   980.0) * h2;

  Ae[8][8][8] = Div(  9.0, 19600.0) * h2;
}


/*  (φ , φ ) - Σ (φ φ , φ _i)U_i の計算  */
```

```
void ecm2(INT ie, INTERVAL_MATRIX & Ge, INTERVAL_MATRIX & D_xxe,
  INTERVAL_MATRIX & D_yye, INTERVAL (*Ae)[9][9],
  VECTOR & U, Ninenodes *ielmt)
{
  INT i, j, k;
  VECTOR Ue(9);

  for (i = 1; i <= 9; i++) {
    j = ielmt[ie-1].node[i-1];
    Ue(i) = U(j);
  }

  Ge = D_xxe + D_yye;

  for (i = 1; i <= 9; i++)
    for (j = 1; j <= 9; j++)
      for (k = 1; k <= 9; k++)
Ge(i, j) += Ae[i-1][j-1][k-1] * Ue(k);
}


/*  Σ (φ , φ _j φ _i)W_iW_j の計算  */
void ecm3(INT ie, INTERVAL_VECTOR & W2e, INTERVAL (*Ae)[9][9],
  INTERVAL_VECTOR & W, Ninenodes *ielmt)
{
  INT i, j, k;
  INTERVAL_VECTOR We(9);


  for (i = 1; i <= 9; i++) {
    j = ielmt[ie-1].node[i-1];
    We(i) = W(j);
  }

  Clear(W2e);

  for (i = 1; i <= 9; i++)
    for (j = 1; j <= 9; j++)
      for (k = 1; k <= 9; k++)
W2e(i) += Abs(Ae[i-1][j-1][k-1]) * Mul(Sup(We(k)), Sup(We(j)));
}


/*  直接剛性法の実行  */
void assem(INT nelmt, INTERVAL_MATRIX & Lm, INTERVAL_MATRIX & Km_x,
   INTERVAL_MATRIX & Km_y, INTERVAL_MATRIX & Dm_xx,
   INTERVAL_MATRIX & Dm_yy, INTERVAL_MATRIX & Dm_yx,
   INTERVAL_MATRIX & Gm, VECTOR & U,
   INTERVAL (*Ae)[9][9], Ninenodes *ielmt)
{
  INT i, j, k, ie, ii, jj, kk;
  INTERVAL_MATRIX Le(9, 9), K_xe(9, 9), K_ye(9, 9), D_xxe(9, 9),
                  D_yye(9, 9), D_yxe(9, 9), Ge(9, 9);

  Clear(Lm);  Clear(Km_x);  Clear(Km_y);  Clear(Dm_xx);  Clear(Dm_yy);
  Clear(Dm_yx);  Clear(Gm);

/*  直接剛性法  */
  ecm(Le, K_xe, K_ye, D_xxe, D_yye, D_yxe, Ae);
                          /*  要素内で生成される行列は各要素毎には変わらない  */
  for (ie = 1; ie <= nelmt; ie++) {
    ecm2(ie, Ge, D_xxe, D_yye, Ae, U, ielmt); /*  Ge( , ) は要素ごとに異なる */
    for (i = 1; i <= 9; i++) {
      ii = ielmt[ie-1].node[i-1];
      for (j = 1; j <= 9; j++) {
jj = ielmt[ie-1].node[j-1];
Lm(ii, jj) += Le(i, j);
Km_x(ii, jj) += K_xe(i, j);
Km_y(ii, jj) += K_ye(i, j);
Dm_xx(ii, jj) += D_xxe(i, j);
Dm_yy(ii, jj) += D_yye(i, j);
Dm_yx(ii, jj) += D_yxe(i, j);
Gm(ii, jj) += Ge(i, j);
```

```
      }
    }
  }
}


/*  行、列ともに Dirichlet 条件を考慮  */
void dirichlet1(INT nnode, INT nbc, INTEGER_VECTOR & ibc,
       INTERVAL_MATRIX & A)
{
  INT i, ii, j;

  for (i = 1; i <= nbc; i++) {
    ii = ibc(i);
    for (j = 1; j <= nnode; j++)
      A(ii, j) = A(j, ii) = 0.0;
    A(ii, ii) = 1.0;
  }
}

/*  列にのみ Dirichlet 条件を考慮  */
void dirichlet2(INT nnode, INT nbc, INTEGER_VECTOR & ibc,
INTERVAL_MATRIX & A)
{
  INT i, ii, j;

  for (i = 1; i <= nbc; i++) {
    ii = ibc(i);
    for (j = 1; j <= nnode; j++)
      A(j, ii) = 0.0;
  }
}


/*  転置行列を作成  */
void Transpose(INT nnode, INTERVAL_MATRIX & A)
{
  INT i, j;
  INTERVAL element;

  for (i = 1; i <= nnode; i++)
    for (j = i+1; j <= nnode; j++) {
      element = A(i, j);
      A(i, j) = A(j, i);
      A(j, i) = element;
    }
}


/* Calculation of (‖ ∇ v_0‖  and ‖ v_h‖ ) */
void v0_norm(INT nnode, INT nbc, INTEGER_VECTOR & ibc,
     INTERVAL_MATRIX & Lm, INTERVAL_MATRIX & Km_x,
     INTERVAL_MATRIX & Km_y, INTERVAL_MATRIX & Dm_xx,
     INTERVAL_MATRIX & Dm_yy, INTERVAL_MATRIX & Dm_yx,
     VECTOR & U, INTERVAL & nabla_v0_norm, INTERVAL & vh_norm)
{
  INT i, j;
  INT done;

  INTERVAL_MATRIX K_x(nnode, nnode);                    /*  K_x = (K~m_x)^T  */
  K_x = Km_x;

  dirichlet2(nnode, nbc, ibc, Km_x);

  INTERVAL_VECTOR Kxu(nnode), U_px(nnode);

  Kxu = Km_x * U;
  done = 0;
  U_px = ILSS(Lm, Kxu, done);                 /*  ∇^-_x u_h の係数ベクトル  */


  INTERVAL_MATRIX K_y(nnode, nnode);                    /*  K_y = (K~m_y)^T  */
  K_y = Km_y;
```

```
  dirichlet2(nnode, nbc, ibc, Km_y);

  INTERVAL_VECTOR Kyu(nnode), U_py(nnode);

  Kyu = Km_y * U;
  done = 0;
  U_py = ILSS(Lm, Kyu, done);                 /*  ∇^-_y u_h の係数ベクトル  */


  INTERVAL_VECTOR U2(nnode);                       /*  u_h^2 の係数ベクトル  */
  for (i = 1; i <= nnode; i++)
    U2(i) = MulBounds(U(i), U(i));
                        /*  実数同士の乗算の結果を包み込むような区間で返す  */

  INTERVAL first_term1, first_term2, first_term3, first_term4;
  INTERVAL second_term1, second_term2, second_term3, second_term4,
           second_term5, second_term6;

  first_term1 = (U_px + U_py) * (Lm *(U_px + U_py));
  first_term2 = -2.0 * U_px * Kxu;
  first_term3 = -2.0 * U_py * Kyu;

  INTERVAL_MATRIX Dm(nnode, nnode);
  Dm = Dm_xx + Dm_yy;
  dirichlet1(nnode, nbc, ibc, Dm);

  first_term4 = U * (Dm * U);


  dirichlet1(nnode, nbc, ibc, Lm);

  second_term1 = U2 * (Lm * U2);

  Transpose(nnode, K_x);
  dirichlet2(nnode, nbc, ibc, K_x);

  second_term2 = 2.0 * U_px * (K_x * U2);

  Transpose(nnode, K_y);
  dirichlet2(nnode, nbc, ibc, K_y);

  second_term3 = 2.0 * U_py * (K_y * U2);

  second_term4 = U_px * (Dm_xx * U_px);
  second_term5 = 2.0 * U_py * (Dm_yx * U_px);
  second_term6 = U_py * (Dm_yy * U_py);


  INTERVAL_VECTOR Lf_Du(nnode), v_h(nnode);
  INTERVAL nabla_vh_norm;

  Lf_Du = Lm * U2 - Dm * U;                                  /*  L・f - D・u  */
  v_h = ILSS(Dm, Lf_Du, done);
  nabla_vh_norm = Sqrt(Abs(v_h * (Dm * v_h)));

  vh_norm = Sqrt(Abs(v_h * (Lm * v_h)));

  nabla_v0_norm =
    Sqrt(first_term1 + first_term2 + first_term3 + first_term4)
      + Ih / Mul(2.0, Pi)
          * Sqrt(second_term1 + second_term2 + second_term3
 + second_term4 + second_term5 + second_term6
)
      + nabla_vh_norm;
}

/*  Calculation of Sup_Norm  */
REAL sup_norm(INT N, INTERVAL_VECTOR & X)
{
  INT i;
  REAL sup, Sup;
```

```
  Sup = 0.0;
  for (i = 1; i <= N; i++) {
    sup = Abs(X(i));
    if (sup > Sup)
      Sup = sup;
  }

  return Sup;
}


main() {
  INT i, j, k, ii, iii, jj, jjj, kk, kkk;
  INT ie;                                                    /*  要素番号  */
  INT N;                                                     /*  分割数-1  */
  INT nelmt;                                                 /*  総要素数  */
  INT nnode;                                                 /*  総節点数  */
  INT nbc;                                          /*  境界条件を課す節点数  */

  cout << setprecision(13);

  cin >> N;
  cout << "分割数 : " << N + 1 << endl;

  Ih = Div(1.0, N + 1);

  cin >> nelmt;
  cin >> nnode;
  cin >> nbc;

  Ninenodes ielmt[nelmt];                       /*  各要素を構成する節点番号  */
  INTEGER_VECTOR ibc(nbc);                       /*  境界条件を課す節点番号  */

  for (i = 0; i < nelmt; i++)
    for (j = 0; j < 9; j++)
      cin >> ielmt[i].node[j];

  cin >> ibc;

  VECTOR U(nnode);                                           /*  近似解  */
  cin >> U;

  INTERVAL_MATRIX Lm(nnode, nnode), Km_x(nnode, nnode), Km_y(nnode, nnode),
                  Dm_xx(nnode, nnode), Dm_yy(nnode, nnode),
                  Dm_yx(nnode, nnode), Gm(nnode, nnode);
  INTERVAL Ae[9][9][9];

/*  直接剛性法による計算  */
  assem(nelmt, Lm, Km_x, Km_y, Dm_xx, Dm_yy, Dm_yx, Gm, U, Ae, ielmt);


  INTERVAL nabla_v0_norm;
  INTERVAL vh_norm;

  v0_norm(nnode, nbc, ibc, Lm, Km_x, Km_y, Dm_xx, Dm_yy, Dm_yx, U,
  nabla_v0_norm, vh_norm);

  dirichlet1(nnode, nbc, ibc, Gm);

  MATRIX Gia(nnode, nnode);
  MATRIX G(nnode, nnode), Ginv(nnode, nnode);

  G = Mid(Gm);
        /*  Inverse( ) は MATRIX 型にのみ有効なので、行列を MATRIX 型にする  */

  Ginv = Inverse(G);                                        /*  G の逆行列  */

/*  G の逆行列の絶対値行列  */
  for (i = 1; i <= nnode; i++)
    for (j = 1; j <= nnode; j++)
      Gia(i, j) = Abs(Ginv(i,j));
```

```
/*  主要部分  */
  INT count;                                                  /*  反復回数  */

  INTERVAL Ialp;                                    /*  α^(i) を区間で計算  */
  INTERVAL T_ort;                                             /*  α^(i+1)  */
  INTERVAL v0_nabla_w_ort_Norm;                  /* ‖v_0‖ + C_0 h α^(i)  */
  INTERVAL nabla_v0_nabla_w_ort_Norm;         /*  ‖∇ v_0‖ + C_0 h α^(i)  */

  INTERVAL_VECTOR Phi(nnode);                /*  ‖φ_i‖ をベクトル形にする  */
  INTERVAL_VECTOR g(nnode);
  INTERVAL_VECTOR T(nnode);                                   /*  |D^(-1)| C  */
  INTERVAL_VECTOR W(nnode);
  INTERVAL_VECTOR UW(nnode);                                  /*  U(i) + W(i)  */
  INTERVAL_VECTOR U2W(nnode);                               /*  2 * U(i) + W(i)  */
  INTERVAL_VECTOR W2(nnode);                        /*  Σ (φ, φ_j φ_i)W_iW_j  */

  INTERVAL_VECTOR W2e(9);              /*  各要素内での (φ, φ_j φ_i)W_iW_j  */

  VECTOR e(nnode);                       /*  ‖φ_i‖_∞ をベクトル形にする  */

  REAL SUP;                                        /*  Calculation of Sup_Norm  */
  REAL WIDTH, width;                                          /*  W_i の幅  */


/*  ‖φ_i‖ の計算  */
  for (i = 1; i <= nnode; i++)
    Phi(i) = Sqrt(Lm(i,i));

/*  ‖φ_i‖_∞ の計算  */
  Initialize(e, 1.0);

/*  ‖φ_i‖, ‖φ_i‖_∞ の Dirichlet 条件の考慮  */
  for (i = 1; i <= nbc; i++) {
    ii = ibc(i);
    Phi(ii) = 0.0;
    e(ii) = 0.0;
  }


/*  反復の初期値  */
  Clear(W);
  Ialp = Hull(0.0);

/*  反復開始  */
  for (count = 1; count <= LOOP; count++) {
    v0_nabla_w_ort_Norm
      = Ih / Mul(2.0, Pi) * nabla_v0_norm + vh_norm + Ih / Mul(2.0, Pi) * Ialp;
    nabla_v0_nabla_w_ort_Norm
      = nabla_v0_norm + Ialp;
    UW = U + W;
    SUP = sup_norm(nnode, UW);

/*  W2 の計算  */
    Clear(W2);
    for (ie = 1; ie <= nelmt; ie++) {
      ecm3(ie, W2e, Ae, W, ielmt);
      for (i = 1; i <= 9; i++) {
ii = ielmt[ie-1].node[i-1];
W2(ii) += W2e(i);
      }
    }
    for (i = 1; i <= nbc; i++) {
      ii = ibc(i);
      W2(ii) = 0.0;
    }

 /*  |g_i| の評価  */
    g = W2 + v0_nabla_w_ort_Norm
      * (Mul(2.0, SUP) * Phi + v0_nabla_w_ort_Norm * e);

    g = SymHull(Abs(g));                          /*  g_i = [ -|g_i|, |g_i|]  */
    T = Gia * g;                                  /*  T_h = |G^(-1)| g_i  */
```

```
    U2W = 2.0 * U + W;

/*  T^⊥ の評価  */
    T_ort = Ih / Mul(2.0, Pi)
      * (Sqrt(Abs(U2W * (Lm * U2W))) * sup_norm(nnode, W)
  + v0_nabla_w_ort_Norm * Mul(2.0, SUP)
  + nabla_v0_nabla_w_ort_Norm * Div(3.0, Sqrt(8.0))
    * Sqrt(v0_nabla_w_ort_Norm * Div(3.0, Sqrt(8.0))
    * nabla_v0_nabla_w_ort_Norm
   )
 );

/*  検証条件  */
    if (T <= W && Sup(T_ort) <= Inf(Ialp)){
      cout << "Iteration Stop --- 反復回数 : " << count << "回" << endl;

      WIDTH = 0.0;  ii = 1;
      for (i = 1; i <= nnode; i++){
if ( i == ibc(ii)) {
  cout << i << "\t" << U(i) + W(i)<< endl;
  ii++;
}
else
  cout << i << "\t" << U(i) + W(i) + nabla_v0_norm + Ialp << endl;


        width = Diam(U(i) + W(i));
if (WIDTH < width)
  WIDTH = width;
      }
      cout << endl;
      cout << "区間 W_i の幅の最大 = " << WIDTH << endl;
      cout << "無限次元部分の半径 ; α  = " << Abs(Ialp) << endl;
      exit(0);
    }

    W = (1.0 + DELTA) * T;
    Ialp = (1.0 + DELTA) * T_ort;
  }

  cout << "検証失敗" << endl;
  exit(0);
}
```

このプログラムの入力で必要な 分割数, 総要素数、総節点数、境界条件を課す節点数、各要素を構成する節点番号、境界条件を課す節点番号は次のようなプログラムで与えられる。

```
/*
 * input-q.c --- NEWTON-q.C の入力ファイル用
 *
 */

#include <stdio.h>
#include <math.h>

main(){
  int i, j, n;
  int node1, node2, node3, node4, node5, node6, node7, node8, node9;
  double h;

  /*  分割数 n の入力  */
  scanf("%d", &n);

  h = 1.0 / n;

  /*  分割数-1, 総要素数, 総節点数, 境界条件を課す節点数  */
  printf("%d\n", n-1);
  printf("%d\n", n * n);
  printf("%d\n", (2 * n + 1) * (2 * n + 1));
  printf("%d\n", 4 * 2 * n);

  /*  各要素に属する節点番号を表示  */
```

```
  for (j = 1; j < 2*n+1; j = j+2)
    for (i = 1; i < 2*n+1; i = i+2){
      node1 = i + (2 * n + 1) * (j - 1);
      node2 = node1 + 1;
      node3 = node1 + 2;
      node4 = i + (2 * n + 1) * j;
      node5 = node4 + 1;
      node6 = node4 + 2;
      node7 = i + (2 * n + 1) * (j + 1);
      node8 = node7 + 1;
      node9 = node7 + 2;

      printf("%d %d %d %d %d %d %d %d %d\n", node1, node2, node3, node4, node5,
     node6, node7, node8, node9);
    }
  printf("\n");

  /*  ディリクレ境界条件を課す節点番号を表示  */
  for (j = 1; j <= 2*n+1; j++)
    printf("%d ", j);
  for (i = 1; i < 2*n; i++)
    printf("%d %d ", (2 * n + 1) * i + 1, (2 * n + 1) * i + 2 * n + 1);
  for (j = 1; j <= 2*n+1; j++)
    printf("%d ", (2 * n + 1) * 2 * n + j);
  printf("\n");

  return 0;
}
```

謝辞

この論文を作成するにあたり、適切な指導・助言をして下さった桂田祐史助教授に深く感謝致します。

参考文献

[1] Adams, R. A., Sobolev spaces, Academic Press, 1975.

[2] Alefeld, G., Introduction to interval computations, Academic Press, 1983.

[3] Grisvard, P., Elliptic problems in nonsmooth domain, Pitman, 1985.

[4] Johnson, C., Schatz, A.H. & Wahlbin, L.B., Crosswind smear and pointwise errors in stream diffusion finite element methods, Math Comp. 49, 1987, pp.25–38.

[5] 菊地文雄, 有限要素法の数理, 培風館, 1994.

[6] Ladyzenskaya, O.A. (藤田宏, 竹下彬訳), 非圧縮粘性流の数学的理論, 産業図書, 1979.

[7] 増田久弥, 非線形数学, 朝倉書店, 1985.

[8] 中尾充宏, 山本野人, 精度保証付き数値計算, 日本評論社, 1998.

[9] Nakao, M.T., A numerical approach to the proof of existence of solutions for elliptic problems, Japan Journal of Applied Mathematics 5, 1988, pp.313–332.

[10] 中尾充宏, 山本野人, a priori 誤差評価定数の計算機による数値評価について, 京都大学数理解析研究所講究録 990, 1997, pp.224–234.

[11] 大石進一, 精度保証付き数値計算, コロナ社, 2000.

[12] Rump, S.M., Solving nonlinear systems with least significant bit accuracy, Computing 29, 1982, pp.183–200.

[13] Rump, S.M., Solving algebraic problems with high accuracy, A new approach to scientific computation, Axademic Press, 1983.

[14] Schultz, M.H.,Spline analysis, Prentice-Hall, 1973.

[15] 杉原正顯, 室田一雄, 数値計算法の数理, 岩波書店, 1994.

[16] Yamamoto, N. & Nakao, M.T., Numerical verifications for solutions to elliptic equations using residual iterations with higher order finite element, Journal of Computational and Applied Mathematics 60, 1995, pp.271–279.