応用数値解析特論 第 6 回

～2 次元 Poisson 方程式に対する有限要素法 (1)

かつらだ まさし
桂田 祐史
https://m-katsurada.sakura.ne.jp/ana2022/

2023 年 5 月 23 日

目次

本日の内容など

- 前回説明した 1 次元 Poisson 方程式に対する有限要素法の C プログラムを読んでみる。
- 2 次元の Poisson 方程式に対する有限要素法で、連立 1 次方程式が具体的にどのようになるか説明する。

4.5 サンプル・プログラム fem1d.c　4.5.1 問題

以下に紹介する C プログラム `fem1d.c` は

`https://m-katsurada.sakura.ne.jp/program/fem/fem1d.c`

に置いてある。現象数理学科 Mac ならば、ターミナルから

```
curl -O https://m-katsurada.sakura.ne.jp/program/fem/fem1d.c
```

で入手できる。コンパイル、実行の仕方はプログラムの先頭部分に注釈として書いてある (`head fem1d.c` で先頭 10 行を見れば分かる)。

このプログラムが対象としている問題は、$f \equiv 1$ で、境界条件は同次、すなわち $\alpha = \beta = 0$ の場合である。具体的に書き下すと

$$-u'' = 1, \quad u(0) = u'(1) = 0. \tag{1}$$

この問題の厳密解は $u(x) = x(2-x)/2$ である。

4.5.2 プログラムの解説

- `main()` を読むと分かるように、最初に
  - `nnode` 総節点数 (the number of nodes)
  - `nelmt` 総要素数 (the number of elements)
  - `nbc` ディリクレ境界にある接点の個数 (1 または 2)
  - `x[]` 節点の座標
  - `ibc` ディリクレ境界にある接点の節点番号

  を決めている。
- 連立 1 次方程式の構成は、関数 `assem()` で行っている (assemblage)。
  その作業内容は 3 つに分かれる。
  1. `am`, `fm` を 0 クリアする。
  2. すべての有限要素について、要素係数行列 `ae`, 要素自由ベクトル `fe` を関数 `ecm()` で計算して (element coefficient matrix)、それぞれ全体係数行列 `am`、全体自由項ベクトル `fm` に算入する。
  3. ディリクレ境界上にある節点に対応する部分を修正する。

4.5.2 プログラムの解説

- 関数 `ecm()` で必要となる事項の復習。$e_k = [x_{k-1}, x_k]$ とすると、

$$A_k = \frac{1}{x_k - x_{k-1}} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}, \quad \boldsymbol{f}_k = \begin{pmatrix} (f, L_0)_{e_k} \\ (f, L_1)_{e_k} \end{pmatrix}$$

  であった。f を 1 次近似することにすれば、

$$\boldsymbol{f}_k \fallingdotseq \frac{x_k - x_{k-1}}{6} \begin{pmatrix} 2f(x_{k-1}) + f(x_k) \\ f(x_{k-1}) + 2f(x_k) \end{pmatrix}.$$

- 前回の授業の説明では、

  A_k を $m+1$ 次正方行列 A_k^* に “拡大して”、$A^* := \sum_{k=1}^{m} A_k^*$ を計算し、その最初の列と最初の行 *(Dirichlet* 境界条件を課す x_0 に対応*)* を削除する。

  ベクトル $\boldsymbol{f}_k$ についても同様 $(\boldsymbol{f}_k^*,\ \boldsymbol{f}^* := \sum_{k=1}^{m} \boldsymbol{f}_k^* + \boldsymbol{g}_m^*)$。右辺に $\alpha \begin{pmatrix} A_{10}^* \\ \vdots \\ A_{m0}^* \end{pmatrix}$ を移項する $(\alpha$ は *Dirichlet* 条件 $u(0) = \alpha$ の $\alpha)$。

  - この問題では $\alpha = 0$ なので移項せず第 0 列を 0 クリアするだけ。
  - 列や行の削除は大変 (大規模コピー？) なので、書き換えることにする。

4.5.3 実験

fem1d.c のコンパイル&実行& gnuplot によるグラフ描画

```
$ cc -o fem1d fem1d.c
$ ./fem1d
nodal values of u (節点での u の値)
   i      u       i      u       i      u
   0   0.000e+00   1   9.500e-02   2   1.800e-01
   3   2.550e-01   4   3.200e-01   5   3.750e-01
   6   4.200e-01   7   4.550e-01   8   4.800e-01
   9   4.950e-01  10   5.000e-01
$ cat fem1d.out
0.000000 0.000000
0.100000 0.095000
0.200000 0.180000
0.300000 0.255000
0.400000 0.320000
0.500000 0.375000
0.600000 0.420000
0.700000 0.455000
0.800000 0.480000
0.900000 0.495000
1.000000 0.500000
$ gnuplot
gnuplot> plot "fem1d.out" with lp, x*(2-x)/2
```

4.5.3 実験

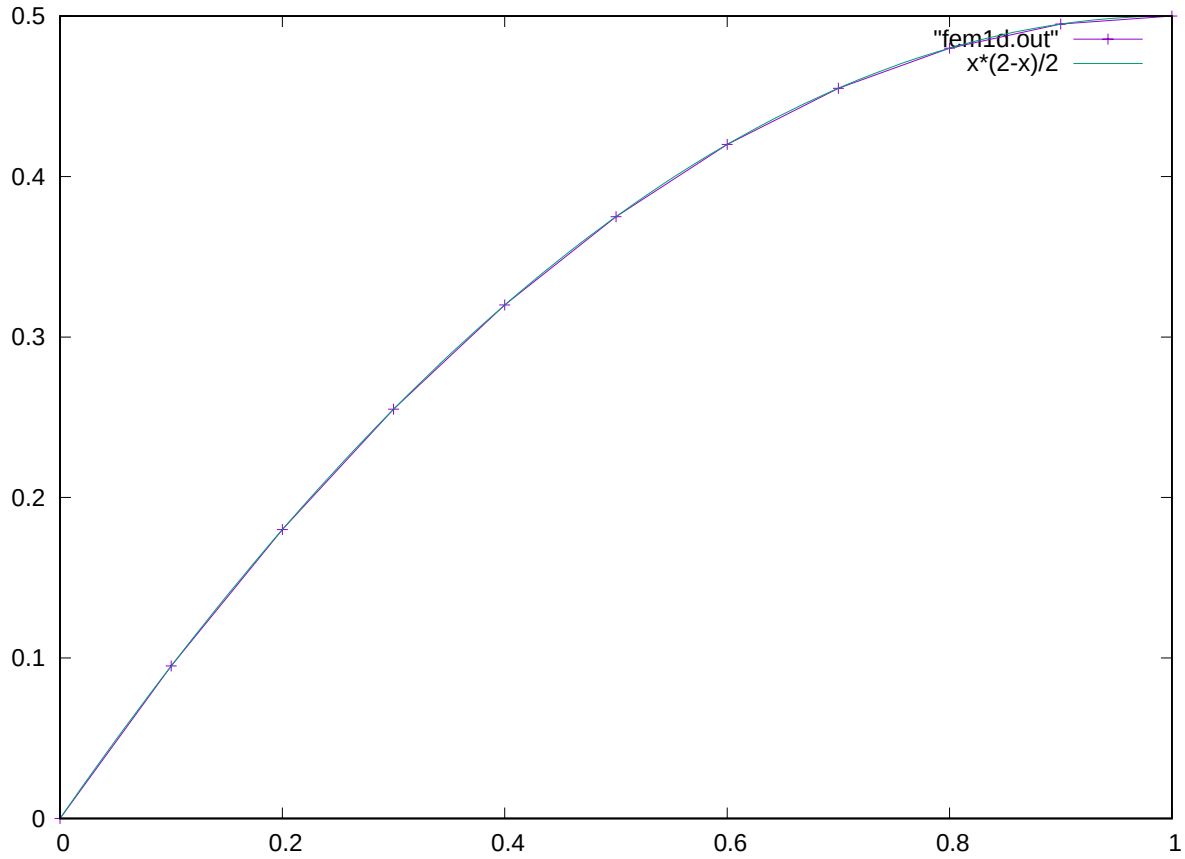


図 1: fem1d.c の計算結果 (m=10) と厳密解 $\frac{x(2-x)}{2}$ のグラフを重ね書き

4.5.4 参考: 昔の練習問題

FreeFem++ がまだなかった頃、有限要素法のプログラムを、C 言語や Fortran のようなプログラミング言語でプログラムを書いていました。

そのときは (アルゴリズムの理解する助けになると考えて) 以下のような練習問題を出していました。参考まで。

1. 両側ディリクレ条件 $u(0) = u(1) = 0$ の問題を解く。
2. 非同次ディリクレ条件 $u(0) = \alpha$ の問題を解く。
3. 非同次 Neumann 条件 $u'(1) = \beta$ の問題を解く。
4. $-(pu')' = f$ という一般の楕円型方程式の問題を解く。
   (p は $\min_x p(x) > 0$ を満たす既知の関数)

5　2 次元の有限要素法

第 2 回の授業で扱った Poisson 方程式の境界値問題を題材に、2 次元領域における有限要素法を説明する (菊地 [?] 第 5 章)。

(思い出す) Ω は $\mathbb{R}^2$ の有界領域で、その境界 Γ は区分的に十分滑らかであるとする。また Γ_1, Γ_2 は条件

$$\Gamma = \overline{\Gamma}_1 \cup \overline{\Gamma}_2, \quad \Gamma_1 \cap \Gamma_2 = \emptyset, \quad \Gamma_1 \neq \emptyset$$

を満たすとする。$f : \Omega \to \mathbb{R}$, $g_1 : \Gamma_1 \to \mathbb{R}$, $g_2 : \Gamma_1 \to \mathbb{R}$ が与えられたとき、次の Poisson 方程式の境界値問題を考える。

問題 (P)

次式を満たす u を求めよ:

$$-\triangle u = f \quad \text{in } \Omega, \tag{2}$$

$$u = g_1 \quad \text{on } \Gamma_1, \tag{3}$$

$$\frac{\partial u}{\partial \boldsymbol{n}} = g_2 \quad \text{on } \Gamma_2, \tag{4}$$

ここで $\boldsymbol{n}$ は Γ の外向き単位法線ベクトルを表す。

大筋は 1 次元の場合と同様だが、(i) 各要素内の計算に面積座標を使うところと、(ii) 直接剛性法が 2 次元でも実現可能であることを理解するところが要点となる。

5.1 三角形要素への分割と区分的 1 次多項式

領域 $\Omega \subset \mathbb{R}^2$ に対し、$\overline{\Omega}$ を三角形 e_k $(1 \leq k \leq N_e)$ に分割する:

$$\overline{\Omega} \fallingdotseq \widehat{\Omega} := \bigcup_{k=1}^{N_e} e_k .$$

ただし、重なりや、すき間、頂点が他の三角形の辺上にあることは避けることにする。

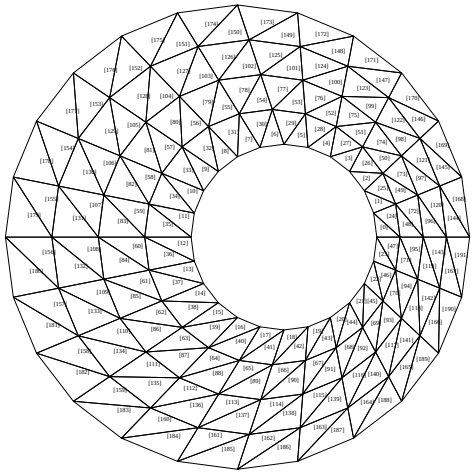


図 2: 二つの円で囲まれた閉領域 $\overline{\Omega}$ を三角形の合併で近似する

Ω が多角形でない限り、境界は「曲がっている」。$\overline{\Omega} = \bigcup_{k=1}^{N_e} e_k$ は期待できない。

各三角形を 三角形 (有限) 要素とよぶ。

5.1 三角形要素への分割と区分的1次多項式

N_e は要素の総数 (the number of elements) で、プログラムでは NELMT のような名前の変数で記憶されることが多い。

有限要素の頂点を**節点** (node) と呼び、$\{\mathrm{P}_i\}_{i=1}^{m}$ のように番号をつけておく。

m は節点の総数 (the number of nodes) で、プログラムでは NNODE のような名前の変数で記憶されることが多い。

注意　1 次元の場合、節点の個数 = 要素の個数 + 1 (NELMT=m+1) という簡単な関係が成立していた。(区間を m 等分したとき、m を用いて、要素を e_k $(1 \leq k \leq m)$, 節点を x_k $(0 \leq k \leq m)$ と番号づけることが出来た。) 2 次元の場合は、そのような関係はない。

5.1 三角形要素への分割と区分的 1 次多項式

$\widehat{\Omega}$ 上連続で、各有限要素 e_k 上で x と y の 1 次多項式関数に等しいものを、**区分的 1 次多項式**と呼び、その全体を $\widetilde{X}$ で表わす ($\widetilde{X}$ はここだけの記号)。

$\{e_k\}_{k=1}^{N_e}$ と $\widetilde{X}$ の対を区分的 1 次有限要素 (piecewise linear finite element) と呼ぶ。

2 変数の 1 次関数 $z = a + bx + cy$ のグラフは平面であるから、$\widetilde{X}$ のグラフは、空間内の三角形を連続につなげた "折れ面" である。

試行関数の空間 $\hat{X}_{g_1}$, 試験関数の空間 $\hat{X}$ は次のように選ぶ。

$$\text{(5)} \quad \hat{X}_{g_1} = \left\{ \hat{w} \in \widetilde{X} \;\middle|\; \hat{w} = \hat{g}_1 \quad \text{on } \Gamma_1 \right\}, \quad \hat{X} = \left\{ \hat{v} \in \widetilde{X} \;\middle|\; \hat{v} = 0 \quad \text{on } \Gamma_1 \right\}.$$

($\hat{g}_1$ は g_1 に近い計算しやすい関数)

$\widetilde{X}$ の基底関数 $\{\phi_i\}_{i=1}^{m}$ は

$$\text{(6a)} \qquad \phi_i \in \widetilde{X} \quad (i = 1, 2, \cdots, m),$$

$$\text{(6b)} \qquad \phi_i(\mathrm{P}_j) = \delta_{ij} \quad (i, j = 1, 2, \cdots, m)$$

満たすものを採用する (この条件で一意に確定し、線形独立であることに注意)。

5.2 三角形 e 上の 1 次関数 L_i と $(L_j, L_i)_e, \langle L_j, L_i \rangle_e$

5.2.1 三角形の面積

平面上の三角形要素 e を考える (本来は、以下の N_i, L_i も含めて、要素によるので、e_k, N_i^k, L_i^k のように書いた方が良いかもしれないが、うっとうしいので k は略する)。

e に属する節点を、反時計まわりに $\mathrm{N}_i = (x_i, y_i)$ $(i = 0, 1, 2)$ とする。

後でしばしば必要になるので、e の面積 $|e|$ を計算しておこう。

$$
\begin{aligned}
|e| &= \frac{1}{2} \det \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix} \\
&= \frac{1}{2} \left[(x_1 - x_0)(y_2 - y_0) - (y_1 - y_0)(x_2 - x_0) \right].
\end{aligned}
$$

1 次多項式全体を P^1 と表す:

$$
P^1 := \left\{ \hat{u} \mid (\exists \alpha_0, \alpha_1, \alpha_2 \in \mathbb{R}) \hat{u}(x, y) = \alpha_0 + \alpha_1 x + \alpha_2 y \right\}.
$$

任意の $\hat{u} \in P^1$ は、3 節点 N_i における値 $u^i := \hat{u}(\mathrm{N}_i)$ $(i = 0, 1, 2)$ を指定すれば定まる ($u_i = \hat{u}(\mathrm{P}_i)$ と混同しないように上に添字をつけた)。これは、直観的に明らかであるが (平面は、その上にある 3 点 (ただし同一直線上にはないとする) を指定すれば定まる)、すぐ後で証明する。

5.2.2 三角形要素の面積座標 L_i

節点 N_i で 1, 他の節点で 0 となる 1 次関数を L_i とする $(i = 0, 1, 2)$。つまり

$$L_i \in P^1, \tag{7a}$$

$$L_i(\mathrm{N}_j) = L_i(x_j, y_j) = \delta_{ij} \quad (j = 0, 1, 2). \tag{7b}$$

(L_0, L_1, L_2) は P^1 の基底になる。実際、1 次独立性はすぐ分かり、$\dim P^1 = 3$ であるから、任意の $\widehat{u} \in P^1$ は、

$$\widehat{u}(x, y) = \sum_{i=0}^{2} u^i L_i(x, y) \quad ((x, y) \in e) \tag{8}$$

と表される。

任意の $\mathrm{P} \in e$ に対して、3 実数 $(L_0(P), L_1(P), L_2(P))$ を P の**面積座標** (area coordinate) あるいは**重心座標** (barycentric coordinate) と呼ぶ。 (色々裏があるけれど今回は駆け足で進む。)

任意の $\mathrm{P} \in e$ に対して次式が成り立つ。

$$L_0(\mathrm{P}) + L_1(\mathrm{P}) + L_2(\mathrm{P}) = 1.$$

(P^1 の基底としては 3 つ必要だが、座標としては 2 つで十分ということになる。)

以下、L_i のグラフの鳥瞰図と等高線を描こう。

Mathematica のコード例と実行結果

```
xs = {0, 3, 1}; ys = {0, 1, 2};
{a, b, c} = Inverse[Transpose[{{1, 1, 1}, xs, ys}]];
L[i_, x_, y_] := a[[i]] + b[[i]]*x + c[[i]]*y
xmin = Min[xs]; xmax = Max[xs]; ymin = Min[ys]; ymax = Max[ys];
gbase = RegionPlot[L[1, x, y] > 0 && L[2, x, y] > 0 && L[3, x, y] > 0,
                   {x, xmin, xmax},{y, ymin, ymax}]
gb=Table[Plot3D[L[i, x, y], {x, xmin, xmax}, {y, ymin, ymax},
              RegionFunction -> Function[{x, y, z},
              L[1, x, y] > 0 && L[2, x, y] > 0 && L[3, x, y] > 0]], {i, 3}]
gc=Table[ContourPlot[L[i, x, y], {x, xmin, xmax}, {y, ymin, ymax},
              RegionFunction -> Function[{x, y, z},
              L[1, x, y] > 0 && L[2, x, y] > 0 && L[3, x, y] > 0]], {i, 3}]
```

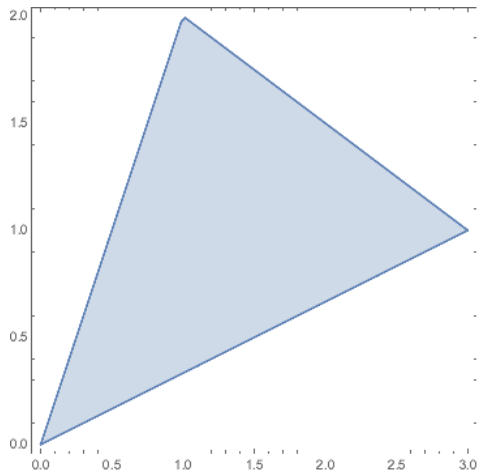


図 3: 三角形要素

Mathematica のコード例と実行結果

授業WWWサイトからブラウザで、あるいは次のようにして 20230523.nb を入手して実行してみよう。

ターミナルで入手して開く

```
curl -O https://m-katsurada.sakura.ne.jp/ana2023/20230523.nb
open 20230523.nb
```

Mathematica のコード例と実行結果

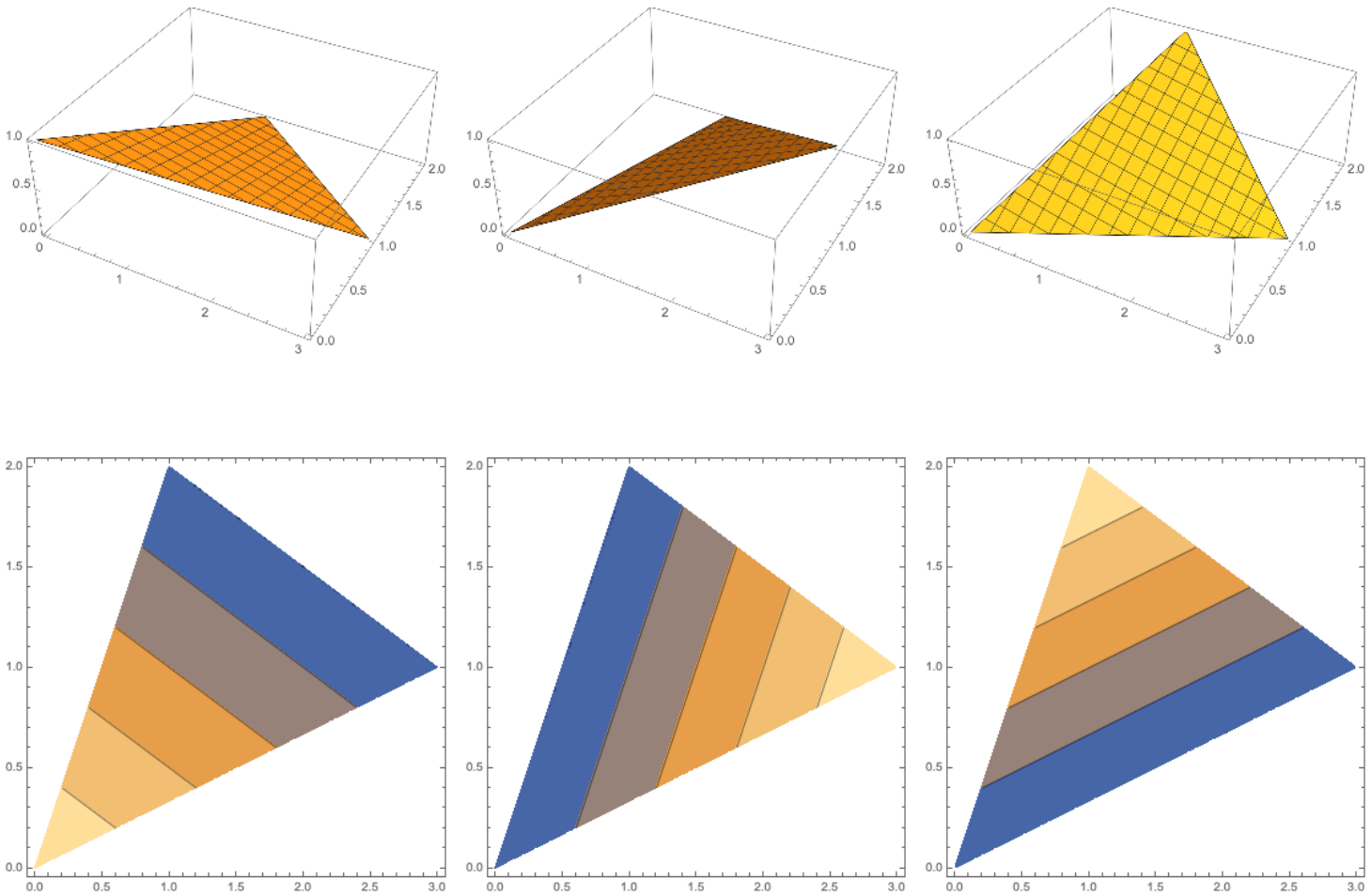


図 4: 左から L_0, L_1, L_2 の鳥瞰図と等高線

5.2.3 面積座標の積の積分と $(L_j, L_i)_e$

面積座標の積の積分については、便利な公式がある。

0 以上の任意の整数 i, j, k に対して

$$\iint_e L_0(x,y)^i L_1(x,y)^j L_2(x,y)^k \, dx\, dy = 2|e| \frac{i!j!k!}{(i+j+k+2)!}. \tag{9}$$

5.2.3 面積座標の積の積分と $(L_j, L_i)_e$

証明　$\mathrm{P}_0 = (0,0)$, $\mathrm{P}_1 = (1,0)$, $\mathrm{P}_2 = (0,1)$ で囲まれる三角形を Δ とし、1 次関数 $\varphi\colon \mathbb{R}^2 \to \mathbb{R}^2$ を

$$\varphi(\mathrm{P}_0) = \mathrm{N}_0, \quad \varphi(\mathrm{P}_1) = \mathrm{N}_1, \quad \varphi(\mathrm{P}_2) = \mathrm{N}_2$$

で定める。このとき

$$\begin{aligned}
&\varphi(\Delta) = e, \\
&\varphi(u,v) = \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}, \\
&\det \varphi'(u,v) = \det \begin{pmatrix} x_1 - x_0 & y_1 - y_0 \\ x_2 - x_0 & y_2 - y_0 \end{pmatrix} = 2|e|.
\end{aligned}$$

ゆえに変数変換 $(x,y) = \varphi(u,v)$ を行なうと

$$\begin{aligned}
\iint_e L_0^i L_1^j L_2^k \, dx\, dy &= \iint_\Delta L_0(\varphi(u,v))^i L_1(\varphi(u,v))^j L_2(\varphi(u,v))^k \, |\det \varphi'(u,v)| \, du\, dv. \\
&= 2|e| \iint_\Delta L_0(\varphi(u,v))^i L_1(\varphi(u,v))^j L_2(\varphi(u,v))^k \, du\, dv.
\end{aligned}$$

5.2.3 面積座標の積の積分と $(L_j, L_i)_e$

$L_i(\varphi(u,v))$ は (1 次関数と 1 次関数の合成であるから)、u, v についての 1 次関数で、

$$L_i(\mathrm{N}_j) = L_i(\varphi(\mathrm{P}_j)) = \delta_{ij}$$

を満たすことから

$$L_0(\varphi(u,v)) = 1-u-v, \quad L_1(\varphi(u,v)) = u, \quad L_2(\varphi(u,v)) = v$$

(各等式の両辺は 1 次関数で、N_j での値が一致するから、全体で一致する).

ゆえに

$$\begin{aligned}\iint_e L_0^i L_1^j L_2^k \, dxdy &= 2|e| \iint_\Delta (1-u-v)^i u^j v^k \, dudv \\ &= 2|e| \int_0^1 u^j \left(\int_0^{1-u} (1-u-v)^i v^k \, dv \right) du.\end{aligned}$$

右辺の内側の積分で、$v = (1-u)t$ $(0 \le t \le 1)$ と変数変換すると

$$dv = (1-u)dt, \quad (1-u-v)^i = ((1-u)-(1-u)t)^i = (1-u)^i(1-t)^i$$

であるから

$$\begin{aligned}\int_0^{1-u} (1-u-v)^i v^k dv &= \int_0^1 (1-u)^i (1-t)^i (1-u)^k t^k \cdot (1-u) dt \\ &= (1-u)^{i+k+1} \int_0^1 (1-t)^i t^k \, dt.\end{aligned}$$

5.2.3 面積座標の積の積分と $(L_j, L_i)_e$

ゆえに

$$\iint_e L_0^i L_1^j L_2^k \, dx\, dy = 2|e| \int_0^1 (1-u)^{i+k+1} u^j \, du \int_0^1 (1-t)^i t^k \, dt$$
$$= 2|e| B(i+k+2, j+1) B(i+1, k+1).$$

ただし B は次式で定義されるベータ関数である:

$$B(p,q) = \int_0^1 (1-t)^{p-1} t^{q-1} \, dt \quad (p,q>0).$$

このベータ関数と、ガンマ関数

$$\Gamma(x) := \int_0^\infty t^{x-1} e^{-t} \, dt \quad (x>0)$$

について、次の有名な公式が成り立つ (証明は例えば桂田 [?] § E.2 の命題 E.2.4)。

$$B(p,q) = \frac{\Gamma(p)\Gamma(q)}{\Gamma(p+q)}, \quad \Gamma(n+1) = n! \quad (n \in \mathbb{N}).$$

ゆえに

$$\iint_e L_0^i L_1^j L_2^k \, dx\, dy = 2|e| \frac{\Gamma(i+k+2)\,\Gamma(j+1)}{\Gamma(i+j+k+3)} \cdot \frac{\Gamma(i+1)\Gamma(k+1)}{\Gamma(i+k+2)}$$
$$= 2|e| \frac{\Gamma(i+1)\Gamma(j+1)\Gamma(k+1)}{\Gamma(i+j+k+3)} = 2|e| \frac{i!j!k!}{(i+j+k+2)!}. \quad \square$$

5.2.3 面積座標の積の積分と $(L_j, L_i)_e$

$i = j$ の場合、それ以外の添字 $(\in \{0,1,2\})$ を k, ℓ とすると

$$(L_j, L_i)_e = \iint_e L_i^2 L_k^0 L_\ell^0 \, dxdy = 2\,|e| \, \frac{2!0!0!}{(2+0+0+2)!} = \frac{1}{6}|e|.$$

$i \neq j$ の場合、それ以外の添字 $(\in \{0,1,2\})$ を k とすると

$$(L_j, L_i)_e = \iint_e L_i^1 L_j^1 L_k^0 \, dxdy = 2\,|e| \, \frac{1!1!0!}{(1+1+0+2)!} = \frac{1}{12}\,|e|\,.$$

5.2.4 $L_i(x,y) = a_i + b_i x + c_i y$ の係数決定と $\langle L_j, L_i \rangle_e$

$$L_i(x,y) = a_i + b_i x + c_i y$$

とおくと

$$\delta_{ij} = L_j(x_i, y_i) = a_j + b_j x_i + c_j y_i = (1 \ x_i \ y_i) \begin{pmatrix} a_j \\ b_j \\ c_j \end{pmatrix}$$

であるから

$$(\heartsuit) \qquad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & x_0 & y_0 \\ 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \end{pmatrix} \begin{pmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ c_0 & c_1 & c_2 \end{pmatrix}.$$

$$A := \begin{pmatrix} 1 & x_0 & y_0 \\ 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \end{pmatrix}$$

とおくと

$$\det A = \det \begin{pmatrix} 1 & x_0 & y_0 \\ 0 & x_1 - x_0 & y_1 - y_0 \\ 0 & x_2 - x_0 & y_2 - y_0 \end{pmatrix} = \begin{vmatrix} x_1 - x_0 & y_1 - y_0 \\ x_2 - x_0 & y_2 - y_0 \end{vmatrix} = 2\,|e| > 0.$$

ゆえに A は逆行列を持つ。

5.2.4 $L_i(x,y) = a_i + b_i x + c_i y$ の係数決定と $\langle L_j, L_i \rangle_e$

(♡) から

$$\begin{pmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ c_0 & c_1 & c_2 \end{pmatrix} = A^{-1}.$$

Cramer の公式によって

$$\begin{aligned} A^{-1} &= \frac{1}{\det A} \begin{pmatrix} (-1)^{1+1}\begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & (-1)^{1+2}\begin{vmatrix} 1 & y_1 \\ 1 & y_2 \end{vmatrix} & (-1)^{1+3}\begin{vmatrix} 1 & x_1 \\ 1 & x_2 \end{vmatrix} \\ (-1)^{2+1}\begin{vmatrix} x_0 & y_0 \\ x_2 & y_2 \end{vmatrix} & (-1)^{2+2}\begin{vmatrix} 1 & y_0 \\ 1 & y_2 \end{vmatrix} & (-1)^{3+3}\begin{vmatrix} 1 & x_0 \\ 1 & x_2 \end{vmatrix} \\ (-1)^{3+1}\begin{vmatrix} x_0 & y_0 \\ x_1 & y_1 \end{vmatrix} & (-1)^{3+2}\begin{vmatrix} 1 & y_0 \\ 1 & y_1 \end{vmatrix} & (-1)^{3+3}\begin{vmatrix} 1 & x_0 \\ 1 & x_1 \end{vmatrix} \end{pmatrix}^\top \\ &= \frac{1}{\det A} \begin{pmatrix} x_1y_2 - y_1x_2 & -(y_2 - y_1) & x_2 - x_1 \\ -(x_0y_2 - y_0x_2) & y_2 - y_0 & -(x_2 - x_0) \\ x_0y_1 - y_0x_1 & -(y_1 - y_0) & x_1 - x_0 \end{pmatrix}^\top \\ &= \frac{1}{\det A} \begin{pmatrix} x_1y_2 - y_1x_2 & x_2y_0 - y_2x_0 & x_0y_1 - y_0x_1 \\ y_1 - y_2 & y_2 - y_0 & y_0 - y_1 \\ x_2 - x_1 & x_0 - x_2 & x_1 - x_0 \end{pmatrix}. \end{aligned}$$

5.2.4 $L_i(x,y) = a_i + b_i x + c_i y$ の係数決定と $\langle L_j, L_i \rangle_e$

A^{-1} の下 2 行 b_k, c_k $(k = 0, 1, 2)$ が得られれば、$\langle L_j, L_i \rangle_e$ が計算できる:

$$\langle L_j, L_i \rangle_e = \iint_e \nabla L_j \cdot \nabla L_i \, dx\, dy = \iint_e \begin{pmatrix} b_j \\ c_j \end{pmatrix} \cdot \begin{pmatrix} b_i \\ c_i \end{pmatrix} dx\, dy \tag{10}$$
$$= (b_j b_i + c_j c_i)|e|.$$

5.3 要素係数行列の計算

「積分は積分範囲を分割して計算し、後から和を取ればよい」ので、弱形式

$$\langle \widehat{u}, \widehat{v} \rangle = (f, \widehat{v}) + [g_2, \widehat{v}] \quad (\widehat{v} \in \widehat{X})$$

は

$$\sum_{k=1}^{N_e} \langle \widehat{u}, \widehat{v} \rangle_{e_k} = \sum_{k=1}^{N_e} (f, \widehat{v})_{e_k} + [g_2, \widehat{v}] \quad (\widehat{v} \in \widehat{X}) \tag{11}$$

となる。ただし

$$\langle \widehat{u}, \widehat{v} \rangle_{e_k} := \int_{e_k} \nabla \widehat{u}(x) \cdot \nabla \widehat{v}(x)\, dx, \quad (f, \widehat{v})_{e_k} := \int_{e_k} f(x)\, \widehat{v}(x)\, dx, \quad [g_2, \widehat{v}] := \int_{\Gamma_2} g_2 \widehat{v}\, d\sigma.$$

そこで $u^j := \widehat{u}(\mathrm{N}_j)$, $v^j := \widehat{v}(\mathrm{N}_j)$ $(j = 0, 1, 2)$ を用いて、

$$\widehat{u} = \sum_{j=0}^{2} u^j L_j, \quad \widehat{v} = \sum_{i=0}^{2} v^i L_i \quad (e_k \text{ 上})$$

と表すと

$$\langle \widehat{u}, \widehat{v} \rangle_{e_k} = \sum_{j=0}^{2} \sum_{i=0}^{2} v^i A_{ij}^{(k)} u^j, \quad (f, \widehat{v})_{e_k} = \sum_{i=0}^{2} v^i f_i^{(k)}, \tag{12}$$

ただし

$$A_{ij}^{(k)} := \langle L_j, L_i \rangle_{e_k}, \quad f_i^{(k)} := (f, L_i)_{e_k}. \tag{13}$$

5.3 要素係数行列の計算

そこで

$$\boldsymbol{u}_k := \begin{pmatrix} u^0 \\ u^1 \\ u^2 \end{pmatrix}, \quad \boldsymbol{v}_k := \begin{pmatrix} v^0 \\ v^1 \\ v^2 \end{pmatrix}, \quad \boldsymbol{f}_k := \begin{pmatrix} f_0^{(k)} \\ f_1^{(k)} \\ f_2^{(k)} \end{pmatrix}, \tag{14a}$$

$$A_k := \begin{pmatrix} A_{00}^{(k)} & A_{01}^{(k)} & A_{02}^{(k)} \\ A_{10}^{(k)} & A_{11}^{(k)} & A_{12}^{(k)} \\ A_{20}^{(k)} & A_{21}^{(k)} & A_{22}^{(k)} \end{pmatrix} \tag{14b}$$

とおくと、 A_k は対称行列で、

$$\langle \widehat{u}, \widehat{v} \rangle_{e_k} = \boldsymbol{v}_k^T A_k \boldsymbol{u}_k, \quad (f, \widehat{v})_{e_k} = \boldsymbol{v}_k^T \boldsymbol{f}_k. \tag{15}$$

(線積分 $[g_2, \hat{v}]$ については、今回の授業では説明を省略する。とりあえず $g_2 = 0$ と考えて授業を聴いて下さい。桂田 [?] には書いてある。)

$A_{ij}^{(k)} = \langle L_j, L_i \rangle_{e_k}$ の計算は (10) で済んでいる。

$f_i^{(k)} = (f, L_i)_{e_k}$ については、例えば L_i を 1 次関数補間して

$$(f, L_i)_{e_k} \fallingdotseq \left(\sum_{j=0}^{2} f(\mathrm{N}_j) L_j, L_i \right)_{e_k} = \sum_{j=0}^{2} f(\mathrm{N}_j) \, (L_j, L_i)_{e_k} \, .$$

のように近似すれば

$$(L_j, L_i)_{e_k} = \begin{cases} |e_k|/6 & (i = j \text{ のとき}), \\ |e_k|/12 & (i \neq j \text{ のとき}). \end{cases}$$

であるから

$$f_0^{(k)} \fallingdotseq \frac{|e_k|}{12} (2f(\mathrm{N}_0) + f(\mathrm{N}_1) + f(\mathrm{N}_2)),$$

$$f_1^{(k)} \fallingdotseq \frac{|e_k|}{12} (f(\mathrm{N}_0) + 2f(\mathrm{N}_1) + f(\mathrm{N}_2)),$$

$$f_2^{(k)} \fallingdotseq \frac{|e_k|}{12} (f(\mathrm{N}_0) + f(\mathrm{N}_1) + 2f(\mathrm{N}_2)).$$

5.4 近似方程式の組み立て — 直接剛性法

$u_i := \widehat{u}(\mathrm{P}_i)$, $v_i := \widehat{v}(\mathrm{P}_i)$ $(i = 1, 2, \cdots, m)$ として、

$$\boldsymbol{u} := \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{pmatrix}, \quad \boldsymbol{v} := \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{pmatrix} \tag{16}$$

とおく。

有限要素 e_k $(k = 1, 2, \cdots, N_e)$ の節点 N_0, N_1, N_2 に対して、

$$\mathrm{N}_0 = \mathrm{P}_{i_{k,0}}, \quad \mathrm{N}_1 = \mathrm{P}_{i_{k,1}}, \quad \mathrm{N}_2 = \mathrm{P}_{i_{k,2}}$$

となる整数 $i_{k,0}$, $i_{k,1}$, $i_{k,2}$ を取る (これらを**全体節点番号**と呼ぶ)。

$\boldsymbol{f}_k^* \in \mathbb{R}^m$ を $i_{k,0}$ 成分 $= f_0^{(k)}$, $i_{k,1}$ 成分 $= f_1^{(k)}$, $i_{k,2}$ 成分 $= f_2^{(k)}$ で、それ以外の成分はすべて 0 であるようなベクトルとする。例えば $i_0^{(k)} < i_1^{(k)} < i_2^{(k)}$ ならば

$$\boldsymbol{f}_k^* = \begin{matrix} 1 \\ \downarrow \end{matrix} \quad \begin{matrix} i_{k,0} \\ \downarrow \end{matrix} \quad \begin{matrix} i_{k,1} \\ \downarrow \end{matrix} \quad \begin{matrix} i_{k,2} \\ \downarrow \end{matrix} \quad \begin{matrix} m \\ \downarrow \end{matrix}$$
$$(0 \quad \cdots 0 \quad f_0^{(k)} \quad 0 \cdots 0 \quad f_1^{(k)} \quad 0 \cdots 0 \quad f_2^{(k)} \quad 0 \cdots \quad 0)^\top.$$

このように $\boldsymbol{f}_k^*$ を定義すると、次が成り立つ。

$$(f, \widehat{v})_{e_k} = \boldsymbol{v}_k^\top \boldsymbol{f}_k = \boldsymbol{v}^\top \boldsymbol{f}_k^* \quad (k = 1, 2, \cdots, N_e). \tag{17}$$

5.4 近似方程式の組み立て — 直接剛性法

同様の考え方で、行列 $A_k^* = (a_{ij}^{(k)})$ を

$$
\begin{aligned}
&a^{(k)}_{i_{k,0}i_{k,0}} = A^{(k)}_{00}, \quad a^{(k)}_{i_{k,0}i_{k,1}} = A^{(k)}_{01}, \quad a^{(k)}_{i_{k,0}i_{k,2}} = A^{(k)}_{02},\\
&a^{(k)}_{i_{k,1}i_{k,0}} = A^{(k)}_{10}, \quad a^{(k)}_{i_{k,1}i_{k,1}} = A^{(k)}_{11}, \quad a^{(k)}_{i_{k,1}i_{k,2}} = A^{(k)}_{12},\\
&a^{(k)}_{i_{k,2}i_{k,0}} = A^{(k)}_{20}, \quad a^{(k)}_{i_{k,2}i_{k,1}} = A^{(k)}_{21}, \quad a^{(k)}_{i_{k,2}i_{k,2}} = A^{(k)}_{22},\\
&\text{それ以外} = 0
\end{aligned}
$$

で定める。例えば $i_{k,0} < i_{k,1} < i_{k,2}$ ならば

$$
A_k^* = \begin{pmatrix}
\begin{matrix} i_{k,0} \\ \downarrow \end{matrix} & \begin{matrix} i_{k,1} \\ \downarrow \end{matrix} & \begin{matrix} i_{k,2} \\ \downarrow \end{matrix} & \\
A^{(k)}_{00} & A^{(k)}_{01} & A^{(k)}_{02} & \leftarrow i_{k,0} \\
A^{(k)}_{10} & A^{(k)}_{11} & A^{(k)}_{12} & \leftarrow i_{k,1} \\
A^{(k)}_{20} & A^{(k)}_{21} & A^{(k)}_{22} & \leftarrow i_{k,2}
\end{pmatrix}
$$

(書いてない成分は 0).

これを用いると

$$
\langle \widehat{u}, \widehat{v} \rangle_{e_k} = \boldsymbol{v}_k^\top A_k \boldsymbol{u}_k = \boldsymbol{v}^\top A_k^* \boldsymbol{u} \quad (k = 1, 2, \cdots, N_e). \tag{18}
$$

5.4 近似方程式の組み立て — 直接剛性法

ゆえに弱形式 (11) は ($g_2 = 0$ と考えている)

$$\sum_{k=1}^{N_e} \boldsymbol{v}^\top A_k^* \boldsymbol{u} = \sum_{k=1}^{N_e} \boldsymbol{v}^\top \boldsymbol{f}_k^*$$

すなわち

$$\boldsymbol{v}^\top \left(\sum_{k=1}^{N_e} A_k^* \boldsymbol{u} - \sum_{k=1}^{N_e} \boldsymbol{f}_k^* \right) = 0$$

と同値になる。

ゆえに

$$\boldsymbol{A}^* := \sum_{k=1}^{N_e} A_k^*, \quad \boldsymbol{f}^* := \sum_{k=1}^{N_e} \boldsymbol{f}_k^*$$

とおけば

(19)
$$\boldsymbol{v}^\top (\boldsymbol{A}^* \boldsymbol{u} - \boldsymbol{f}^*) = 0.$$

5.4 近似方程式の組み立て — 直接剛性法

ここで $\boldsymbol{v}$ は

$$Y := \{ \begin{pmatrix} v_1 \\ \vdots \\ v_m \end{pmatrix} \in \mathbb{R}^m;\ v_i = 0 \quad (\mathrm{P}_i \in \Gamma_1 \text{ なる } i) \}$$

の任意の元であるから、(19) は次と同値である。

$$\boldsymbol{A}^* \boldsymbol{u} - \boldsymbol{f} \in Y^\perp = \{ \begin{pmatrix} w_1 \\ \vdots \\ w_m \end{pmatrix} \in \mathbb{R}^m;\ w_i = 0 \quad (\mathrm{P}_i \notin \Gamma_1 \text{ なる } i) \}$$

すなわち

(20)
$$A^{**} \boldsymbol{u} = \boldsymbol{f}^{**}.$$

ここで

$A^{**} := \boldsymbol{A}^*$ の第 i 行 $(\mathrm{P}_i \in \Gamma_1$ なる $i)$ を除いた行列,

$\boldsymbol{f}^{**} := \boldsymbol{f}^*$ の第 i 成分 $(\mathrm{P}_i \in \Gamma_1$ なる $i)$ を除いた縦ベクトル.

5.4 近似方程式の組み立て — 直接剛性法

$$I := \{i \in \mathbb{N} \mid 1 \leq i \leq m\}, \quad I_1 := \{i \in I \mid P_i \in \Gamma_1\}$$

とおく (I_1 は Γ_1 上にある節点の節点番号全体の集合)。

条件

$$u_i = g_1(\mathrm{P}_i) \quad (i \in I_1)$$

があるから、これを代入して u_i $(i \in I_1)$ を消去できる。以下それを実行する。

A^{**} を列ベクトルで

$$A^{**} = (\boldsymbol{a}_1 \boldsymbol{a}_2 \cdots \boldsymbol{a}_m)$$

のように表示すると、(20) は

(21)
$$\sum_{i=1}^{m} \boldsymbol{a}_i u_i = \boldsymbol{f}^{**}.$$

左辺の $\sum_{i=1}^{m}$ を $\sum_{i \in I \setminus I_1} + \sum_{i \in I_1}$ と分解して、移項すると

$$\sum_{i \in I \setminus I_1} \boldsymbol{a}_i u_i = \boldsymbol{f}^{**} - \sum_{i \in I_1} \boldsymbol{a}_i u_i.$$

5.4 近似方程式の組み立て — 直接剛性法

ゆえに

$$A\boldsymbol{u}^* = \boldsymbol{f},$$

ただし

$$\begin{aligned}
&\boldsymbol{u}^* := \boldsymbol{u} \text{ の第 } i \text{ 成分 } (i \in I_1) \text{ を除いた縦ベクトル,} \\
&A := A^{**} \text{ の第 } i \text{ 列 } (i \in I_1) \text{ を除いた正方行列,} \\
&\boldsymbol{f} := \boldsymbol{f}^{**} - \sum_{i \in I_1} \boldsymbol{a}_i u_i .
\end{aligned}$$

実際にプログラムを作成するとき、A や $\boldsymbol{f}$ が容易に求められることは次回解説する。

5.5 連立１次方程式の具体例

簡単な問題に対する有限要素法の連立１次方程式を実際に求めてみよう。

$$\Omega = (0,1) \times (0,1),$$
$$\Gamma_1 = \{(x,y) \mid x = 0, 0 \le y \le 1\} \cup \{(x,y) \mid 0 \le x \le 1, y = 0\},$$
$$\Gamma_2 = \{(x,y) \mid x = 1, 0 < y \le 1\} \cup \{(x,y) \mid 0 < x \le 1, y = 1\},$$
$$g_1 \equiv 0, \quad g_2 \equiv 0, \quad f \equiv \text{定数関数 } \overline{f}.$$

すなわち

$$-\triangle u = \overline{f} \quad (\text{in } \Omega), \quad u = 0 \quad \text{on } \Gamma_1, \qquad \frac{\partial u}{\partial \boldsymbol{n}} = 0 \quad \text{on } \Gamma_2.$$

5.5 連立 1 次方程式の具体例

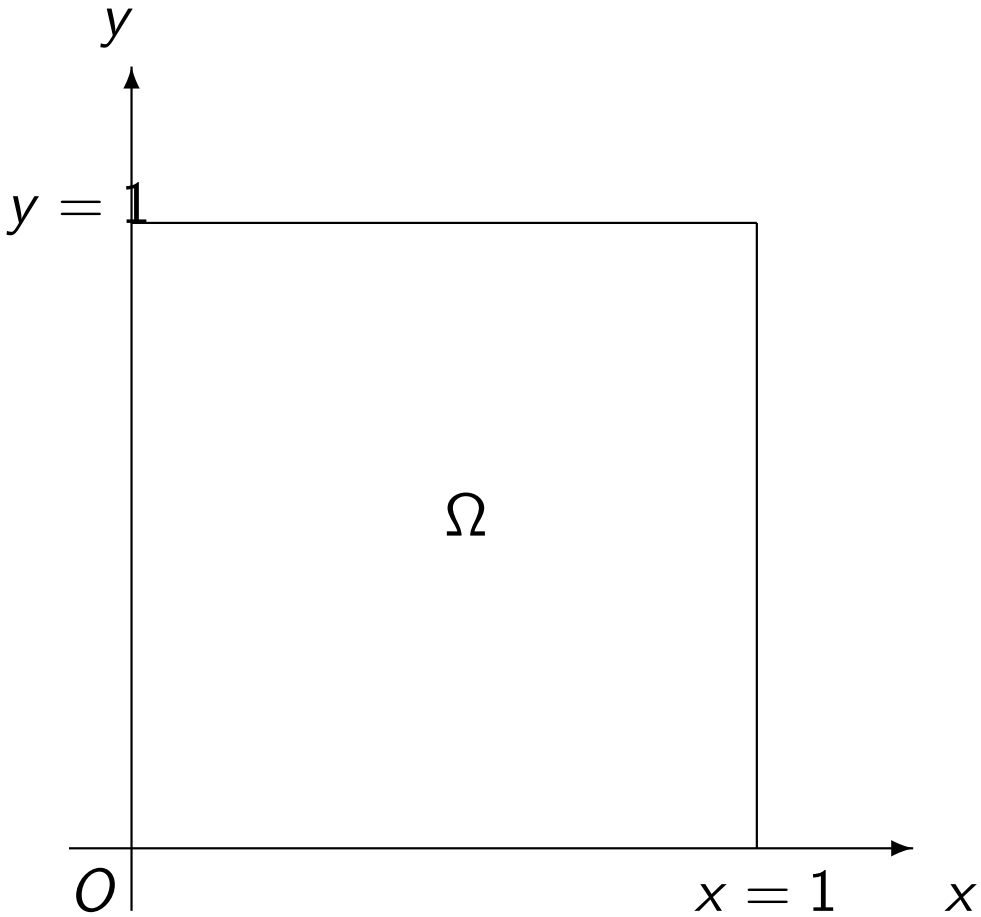


図 5: 領域 Ω

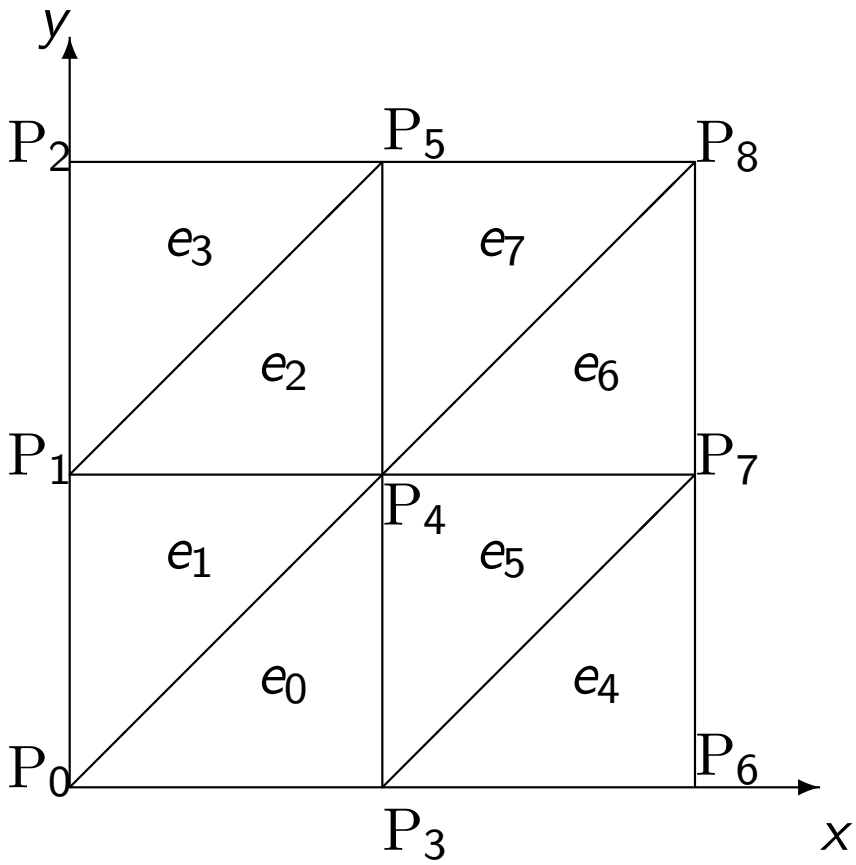


図 6: 要素分割

5.5 連立 1 次方程式の具体例

正方形領域 Ω を図 2 のように 3 角形要素によって要素分割する。

有限要素は次の二つのタイプがある (タイプ I, II と呼ぶことにする)。各々に図 3 のように局所節点番号をつける (左下から反時計回り)。

タイプ I e_0, e_2, e_4, e_6.
タイプ II e_1, e_3, e_5, e_7.

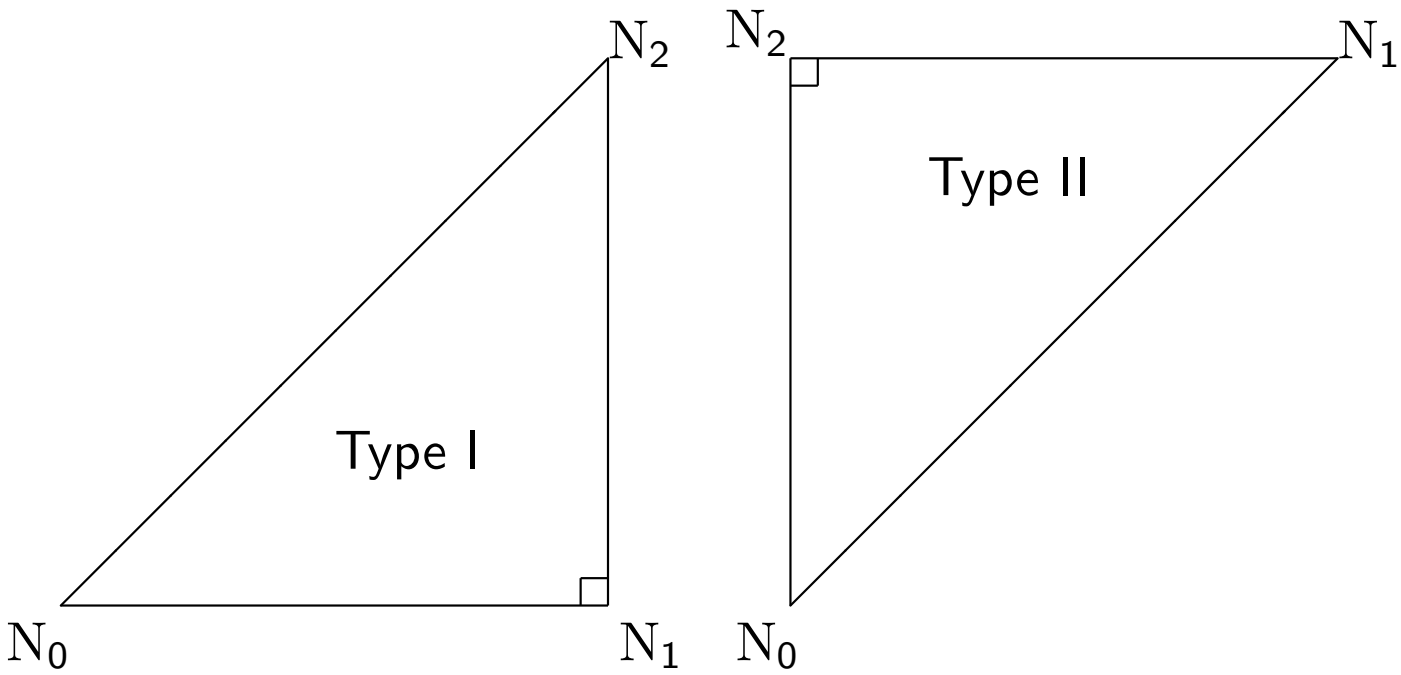


図 7: 二つのタイプの有限要素と局所節点番号

5.5 連立 1 次方程式の具体例

タイプが同じならば、要素係数行列 A_k, 要素自由項ベクトル $\boldsymbol{f}_k$ が等しいことはすぐ分かる。それぞれ A_{I}, A_{II}, f_{I}, f_{II} で表すことにする。

タイプ I については

$$D = h^2, \quad S = \frac{h^2}{2},$$

$$A_I = \frac{1}{2}\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad \boldsymbol{f}_I = \frac{\overline{f}h^2}{6}\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

タイプ II については

$$D = h^2, \quad S = \frac{h^2}{2},$$

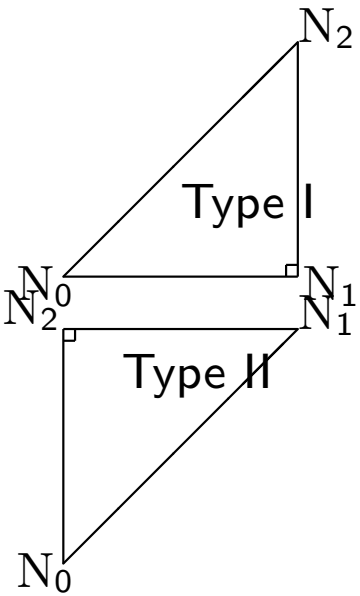
$$A_{II} = \frac{1}{2}\begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ -1 & -1 & 2 \end{pmatrix}, \quad \boldsymbol{f}_{II} = \frac{\overline{f}h^2}{6}\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

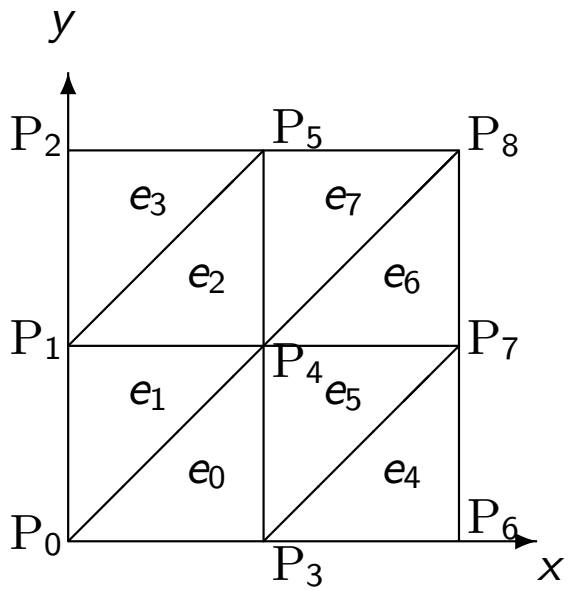
5.5 連立 1 次方程式の具体例

これらから、全体的な近似方程式を作ろう。

そのために局所的な節点番号と、全体的な節点番号の対応づけが必要である。そこで以下のような対応表を用意する (スライド見る場合も手で写すことを推奨 — 要素タイプは不要)。

要素	e_0	e_1	e_2	e_3	e_4	e_5	e_6	e_7
要素タイプ	I	II	I	II	I	II	I	II
N_0 の全体節点番号	0	0	1	1	3	3	4	4
N_1 の全体節点番号	3	4	4	5	6	7	7	8
N_2 の全体節点番号	4	1	5	2	7	4	8	5





5.5 連立 1 次方程式の具体例

これから Galerkin 法の弱形式は

$$\boldsymbol{v}^\top \frac{1}{2} \begin{pmatrix} 2 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -2 & 0 & -1 & 0 & 0 \\ 0 & -2 & 0 & -2 & 8 & -2 & 0 & -2 & 0 \\ 0 & 0 & -1 & 0 & -2 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & -2 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \end{pmatrix} = \boldsymbol{v}^\top \frac{\overline{f} h^2}{6} \begin{pmatrix} 2 \\ 3 \\ 1 \\ 3 \\ 6 \\ 3 \\ 1 \\ 3 \\ 2 \end{pmatrix}$$

for

$$\forall \boldsymbol{v} \in \left\{ (v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8)^\top \in \mathbb{R}^9 \;\middle|\; v_0 = v_1 = v_2 = v_3 = v_6 = 0 \right\}.$$

5.5 連立 1 次方程式の具体例　Dirichlet 境界条件の考慮

$P_i \in \Gamma_1$ となる i について (今の場合 $i = 0, 1, 2, 3, 6$)、第 i 行は削除してよい。

$$\frac{1}{2}\begin{pmatrix} 0 & -2 & 0 & -2 & 8 & -2 & 0 & -2 & 0 \\ 0 & 0 & -1 & 0 & -2 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -2 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \end{pmatrix} = \frac{\overline{f}h^2}{6} \begin{pmatrix} 6 \\ 3 \\ 3 \\ 2 \end{pmatrix}.$$

また $P_i \in \Gamma_1$ となる i について、$u_i = g_1(P_i)$ $(= 0)$. これを代入して移項すると

$$\frac{1}{2}\begin{pmatrix} 8 & -2 & -2 & 0 \\ -2 & 4 & 0 & -1 \\ -2 & 0 & 4 & -1 \\ 0 & -1 & -1 & 2 \end{pmatrix} \begin{pmatrix} u_4 \\ u_5 \\ u_7 \\ u_8 \end{pmatrix} = \frac{\overline{f}h^2}{6} \begin{pmatrix} 6 \\ 3 \\ 3 \\ 2 \end{pmatrix} + \begin{pmatrix} g_1(P_1) + g_1(P_3) \\ g_1(P_2)/2 \\ g_1(P_6)/2 \\ 0 \end{pmatrix}.$$

すなわち

$$\begin{pmatrix} 4 & -1 & -1 & 0 \\ -1 & 2 & 0 & -1/2 \\ -1 & 0 & 2 & -1/2 \\ 0 & -1/2 & -1/2 & 1 \end{pmatrix} \begin{pmatrix} u_4 \\ u_5 \\ u_7 \\ u_8 \end{pmatrix} = \begin{pmatrix} \overline{f}h^2 \\ \overline{f}h^2/2 \\ \overline{f}h^2/2 \\ \overline{f}h^2/3 \end{pmatrix}.$$

5.5 連立 1 次方程式の具体例　Dirichlet 境界条件の考慮

上のやり方では、係数行列とベクトルが “縮小” される。いくつか留意すべき点:

- 例えば MATLAB では、$(0, 1, 2, 3, 6)$, $(4, 5, 7, 8)$ という添字ベクトルを用意すれば、全体係数行列と全体自由項ベクトルを求めるのは (コーディング上は) 容易である。
- 自分で疎行列を扱うコードを書いていたりする場合はそれなりに面倒。
- データの移動にも計算コストがかかる。

Dirichlet 境界条件の処理には、他のやり方 (行列とベクトルを縮小しない方法) もある。

5.5 連立 1 次方程式の具体例　Dirichlet 境界条件の考慮

ベクトル、行列の縮小を避ける方法 (1)

$P_i \in \Gamma_1$ となる i (この例では $i = 0, 1, 2, 3, 6$) に対して

- i 番目の方程式 ($\boldsymbol{v}^\top$ のせいで $\boldsymbol{0}^\top \boldsymbol{u} = 0$) を $u_i = g_1(P_i)$ で置き換える。
  (結果的に係数行列の第 i 行は $\boldsymbol{e}_i^\top$ で置き換える)

とすることで $\boldsymbol{v}^\top$ が外せる。

$$
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\
0 & 0 & -1/2 & 0 & -1 & 2 & 0 & 0 & -1/2 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & -1/2 & 2 & -1/2 \\
0 & 0 & 0 & 0 & 0 & -1/2 & 0 & -1/2 & 1
\end{pmatrix}
\begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \end{pmatrix}
=
\begin{pmatrix}
g_1(\mathrm{P}_0) \\ g_1(\mathrm{P}_1) \\ g_1(\mathrm{P}_2) \\ g_1(\mathrm{P}_3) \\ \overline{f} h^2 \\ \overline{f} h^2/2 \\ g_1(\mathrm{P}_6) \\ \overline{f} h^2/2 \\ \overline{f} h^2/3
\end{pmatrix}
$$

これは正しい方程式であるが、係数行列が対称でなくなっている (数値計算で不利)。

そこで、係数行列の i 列に 0 でない非対角要素があれば、それと $g_1(P_i)$ との積を右辺に移項する。

(その結果は次のスライド)

5.5 連立 1 次方程式の具体例　Dirichlet 境界条件の考慮

$$
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 4 & -1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & -1 & 2 & 0 & 0 & -1/2 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 2 & -1/2 \\
0 & 0 & 0 & 0 & 0 & -1/2 & 0 & -1/2 & 1
\end{pmatrix}
\begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \end{pmatrix}
$$

$$
= \begin{pmatrix}
g_1(\mathrm{P}_0) \\ g_1(\mathrm{P}_1) \\ g_1(\mathrm{P}_2) \\ g_1(\mathrm{P}_3) \\ \overline{f}h^2 \\ \overline{f}h^2/2 \\ g_1(\mathrm{P}_6) \\ \overline{f}h^2/2 \\ \overline{f}h^2/3
\end{pmatrix}
+ \begin{pmatrix}
0 \\ 0 \\ 0 \\ 0 \\ g_1(P_1) + g_1(P_3) \\ g_1(P_2)/2 \\ 0 \\ g_1(P_6)/2 \\ 0
\end{pmatrix}.
$$

5.5 連立 1 次方程式の具体例　Dirichlet 境界条件の考慮

(説明のため、Galerkin 法の弱形式を再度掲示)

$$\boldsymbol{v}^\top \frac{1}{2} \begin{pmatrix} 2 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -2 & 0 & -1 & 0 & 0 \\ 0 & -2 & 0 & -2 & 8 & -2 & 0 & -2 & 0 \\ 0 & 0 & -1 & 0 & -2 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & -2 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \end{pmatrix} = \boldsymbol{v}^\top \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \end{pmatrix}$$

for

$$\forall \boldsymbol{v} \in \left\{ (v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8)^\top \in \mathbb{R}^9 \;\middle|\; v_0 = v_1 = v_2 = v_3 = v_6 = 0 \right\}.$$

5.5 連立 1 次方程式の具体例　Dirichlet 境界条件の考慮

ベクトル、行列の縮小を避ける方法 (2) (FreeFem++で採用？)

$P_i \in \Gamma_1$ となる i に対して、行列の (i,i) 成分を “terrible great value” tgv $(= 10^{30})$ で置き換え、右辺のベクトルの第 i 成分を tgv $\times\, g_1(P_i)$ で置き換える。方程式が近似方程式に置き換わってしまうが、以下の利点がある。

- 解は実質的にほぼ変わらない (演算精度を 10 進 16 桁弱と想定してる)。
- 行列、ベクトルの縮小 (サイズ変更) は不要。
- 係数行列の対称性は保たれる。
- コーディングの負担 (手間) が少ない。

5.5 連立 1 次方程式の具体例　Dirichlet 境界条件の考慮

T を tgv ($= 10^{30}$) として

$$\begin{pmatrix} T & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & T & -1 & 0 & -2 & 0 & 0 & 0 & 0 \\ 0 & -1 & T & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & T & -2 & 0 & -1 & 0 & 0 \\ 0 & -2 & 0 & -2 & 8 & -2 & 0 & -2 & 0 \\ 0 & 0 & -1 & 0 & -2 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & T & -1 & 0 \\ 0 & 0 & 0 & 0 & -2 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \end{pmatrix} = \begin{pmatrix} T g_1(P_0) \\ T g_1(P_1) \\ T g_1(P_2) \\ T g_1(P_3) \\ f_4 \\ f_5 \\ T g_1(P_6) \\ f_7 \\ f_8 \end{pmatrix}$$

(実は、この方法が数値計算的にも妥当なものであるか、私自身は納得できていないところがある (行列の条件数が大きくなるが大丈夫？)。)

有限要素解を計算する (連立 1 次方程式を作る) ため、何が必要かまとめる。

上の例 ($f \equiv \overline{f}$ (定数), $g_1 \equiv 0$, $g_2 \equiv 0$) では

- 節点の座標 ($i = 1, \cdots, m$ に対して P_i の座標 (x_i, y_i))
- Γ_1 上にある節点の全体節点番号
- 各要素 e_k を構成する節点の全体節点番号 $i_{k,0}$, $i_{k,1}$, $i_{k,2}$

が必要になった。

一般の問題では、次のものも必要になる。

(a) Ω に属する節点 P_i での f の値 $f(P_i)$

(b) Γ_1 上にある節点での g_1 の値

(c) Γ_2 上にある節点の全体節点番号

(d) Γ_2 上にある節点での g_2 の値

以上の情報があれば、Poisson 方程式の境界値問題を解くための一般的な方程式が作成できる。

(Ω, Γ_1, Γ_2, $\{e_k\}$, $\{P_i\}$, f, g_1, g_2 などの情報は、プログラムの中に埋め込まずに入力データとして与えることが出来る。)

5.6.2 サンプルプログラム紹介

菊地 [?] にはサンプル・プログラム (FORTRAN, C 言語) も用意されている。

[?] の初版の FORTRAN プログラムを、移植した C 言語プログラムを紹介する。長いので別資料として紹介する。

1. 百聞は一見しかず。まず実行例を見てもらう。
2. プログラムが何をするか、入力と出力を理解する。
   有限要素解を求めるプログラム (`naive`, `band`) では、領域や三角形分割の情報を入力データとする。そのため一般性が高くなっている。
3. naive と band の比較をする。数学的にはやること同じ。効率の違いは？
4. プログラムの心臓部分 `assem()` と `ecm()` の解読 (説明したことの確認)。

6.2 試しに実行

参考　授業 WWW サイトの「有限要素法のサンプル C プログラム」

入手、展開、ファイル名確認

```
curl -O https://m-katsurada.sakura.ne.jp/program/fem/fem-mac-20221031.tar.gz
tar xzf fem-mac-20221031.tar.gz
cd fem-mac-20221031
ls
```

とりあえず動作チェック (実行には、cc, ccg (あるいは cglsc), make 等が必要)

コンパイル&テスト

`make`	プログラムのコンパイル
`make test1`	naive の動作確認 (辺を 2,4,8 分割したときの有限要素解の数値データ)
`make test2`	band の動作確認 (辺を 2,4,8 分割したときの有限要素解の数値データ)
`make test3`	band の動作確認 (辺を 2,4,8,16,32 分割したときの有限要素解の等高線表示) 等高線を描いたウィンドウをクリックすると次を表示

途中で引っかかった場合、相談して下さい。

もしかすると、今の院生の Mac には ccg がインストールされていないかも。その場合は `make test3` は実行できない。

ソースプログラム `naive.c`, `band.c` は、それぞれ 321 行、397 行である。

6.3 有限要素解を求めるプログラム naive, band の理解

2 次元多角形領域 Ω における Poisson 方程式の同次 Dirichlet, Neumann 境界値問題

$$-\triangle u(x,y) = f(x,y) := 1 \quad ((x,y) \in \Omega), \tag{22}$$

$$u(x,y) = g_1(x,y) := 0 \quad ((x,y) \in \Gamma_1), \tag{23}$$

$$\frac{\partial u}{\partial \boldsymbol{n}}(x,y) = g_2(x,y) := 0 \quad ((x,y) \in \Gamma_2) \tag{24}$$

を有限要素法で解くプログラムである。

Q ここで Ω, Γ_1, Γ_2 は何か？

A 実は Ω, Γ_1, Γ_2 についてはデータとして入力する。

naive, band ともに、任意の領域&境界についての計算ができる。

(f, g_1, g_2 については、簡単のため、特殊な値 1, 0, 0 が仮定されている。これを一般化するのは適度の演習問題である。)

6.3 有限要素解を求めるプログラム naive, band の理解

入力データの例 `input.dat`

```
9     8     5
0.0         0.0
0.0         0.5
0.0         1.0
0.5         0.0
0.5         0.5
0.5         1.0
1.0         0.0
1.0         0.5
1.0         1.0
 0     3     4     0     4     1
 1     4     5     1     5     2
 3     6     7     3     7     4
 4     7     8     4     8     5
 0     1     2     3     6
```

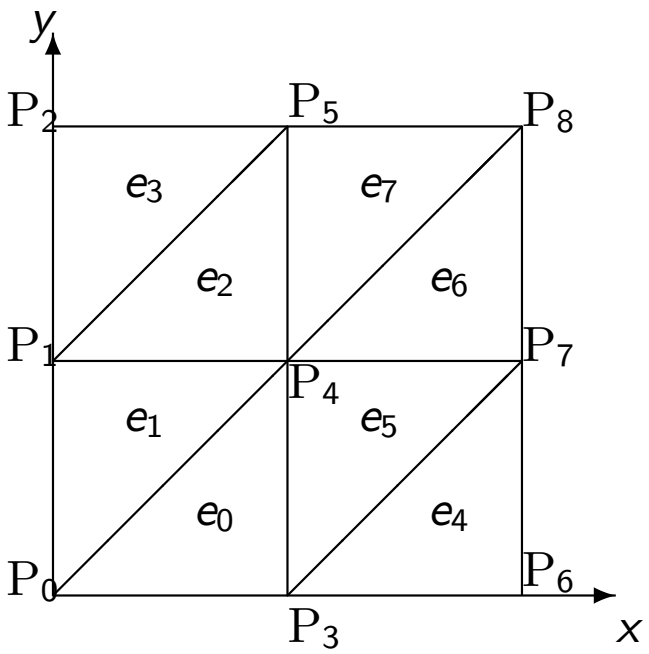


図 8: 要素分割 (各辺を 2 等分してから要素分割)

1. 1 行目には、節点数 (nnode)、要素数 (nelmt)、Γ_1 に属している節点数 (nbc)
2. 2〜10 行は、節点の座標 (x_i, y_i) $(i = 0, 1, \cdots, \mathtt{nnode} - 1)$
3. 11〜14 行は、各要素を構成する節点の全体節点番号 (0 から nelmt−1 までの通し番号)
   節点は各要素を左回りに回るように順序付けてある。
4. 最後に Γ_1 に属する節点の全体節点番号 (nbc 個の番号)

6.3 有限要素解を求めるプログラム naive, band の理解

この形式のデータがあれば、図が描ける (幾何的状況が分かる) ことを理解しよう。

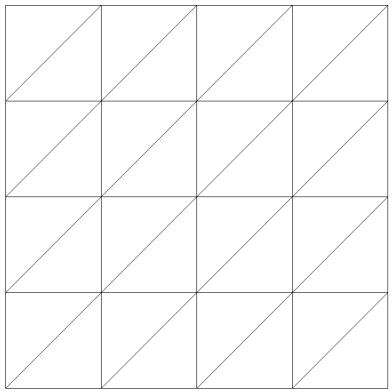
三角形と (結果として)Ω を描く

```
./disp-glsc3d input.dat
./disp-glsc3d input4.dat
cat input4.dat | ./disp-glsc3d
./make-input | ./disp-glsc3d
```

(最後のコマンドに対して、辺を何等分するか、数値 (例えば 64 とか) を入力しよう。)

コマンド 1 | コマンド 2 でコマンド 1 の出力をコマンド 2 に入力できる (パイプ機能)。

`disp-glsc3d` は上の形式のデータを図示するプログラム、`make-input` は正方形領域に対して上の形式のデータを作成するプログラムである。



6.3 有限要素解を求めるプログラム naive, band の理解

naive, band は上の形式の入力データから、有限要素解を計算するプログラム。

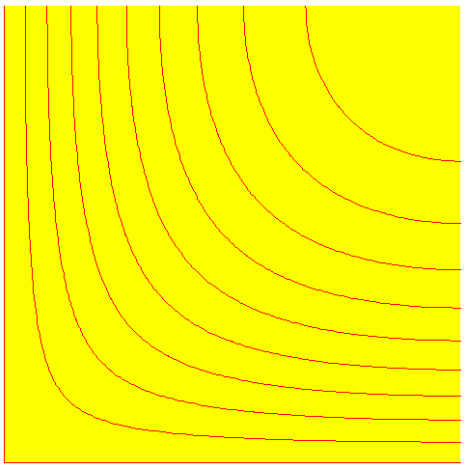
両者は同じ計算を行う。連立 1 次方程式の係数行列が**帯行列** (band matrix) であることを利用して、計算の効率化の工夫をしたのが band で、それをしないのが naive である。

一辺 64 分割で解き比べ (CPU 時間計測), 解の等高線表示

```
echo 64 | ./make-band-input > input64.dat
./disp-glsc3d input64.dat
time ./naive input64.dat
time ./band input64.dat
```

あるマシンで 17.5 秒 vs 0.02 秒　(naive は実際的ではない, ちなみに節点数 4225)

```
./contour-glsc3d band.out
```



6.4 プログラム naive の内部構造

主な関数には以下のようなものがある。

`main()`	
`input()`	入力データ読み込み
`assem()`	全体係数行列 A, 全体自由項ベクトル f の計算 (直接剛性法)
`ecm()`	要素係数行列 A_e, 要素自由項ベクトル f_e の計算
`solve()`	
`output()`	節点パラメーター (節点での解の値) を出力
`f()`	Poisson 方程式 $-\triangle u = f$ の右辺の既知関数 f

主な変数名

`nnode`	節点の総数
`nelmt`	有限要素の総数
`nbc`	Γ_1 (Dirichlet 境界条件を課す) 上の節点の総数
`x[nnode], y[nnode]`	節点の座標
`ielmt[nelmt].node[3]`	各有限要素を構成する節点の番号
`ibc[nbc]`	基本境界条件を課す節点の番号
`am[][]`	全体係数行列
`fm[]`	全体自由項ベクトル

6.4 プログラム naive の内部構造　assem()

assem() は連立 1 次方程式を組み立てる関数。

$A^* := \sum_{k=0}^{N_e-1} A_k^*$ と $\boldsymbol{f}^* := \sum_{k=0}^{N_e-1} \boldsymbol{f}_k^*$ を次のように計算する。

```
/* assemblage of total matrix and vector; */
    for (k = 0; k < nelmt; k++) {
        ecm(k, ielmt, x, y, ae, fe);
        for (i = 0; i < 3; i++) {
            ii = ielmt[k].node[i];
            fm[ii] += fe[i];
            for (j = 0; j < 3; j++) {
                jj = ielmt[k].node[j];
                am[ii][jj] += ae[i][j];
            }
        }
    }
```

ielmt[k].node[i] は、要素 e_k の、局所節点番号が i の節点 N_i の全体節点番号

6.4 プログラム naive の内部構造　ecm()

ecm() は要素係数行列 A_k、要素自由項ベクトル $\boldsymbol{f}_k$ を求める関数。

```
/* 節点の座標を求める */
for (i = 0; i < 3; i++) {
    j = ielmt[k].node[i];
    xe[i] = x[j];
    ye[i] = y[j];
}
```

節点の座標さえ求まれば、A_k, $\boldsymbol{f}_k$ の成分は公式に従って計算するだけである。(それを確かめたければ、naive.c あるいは band.c を見よ。)

6.5 参考課題

以前、FreeFem++ が使えなかった頃は、授業で次のような課題を出していた。

私は「百見は一験にしかず」と考えていて、次のような実験をすることは有益と思っているが、この科目では要求しない。

(a) このプログラムで解ける問題は、境界条件が同次境界条件

$$u = 0 \quad \text{on } \Gamma_1, \qquad \frac{\partial u}{\partial \boldsymbol{n}} = 0 \quad \text{on } \Gamma_2$$

であるが、これを非同次境界条件

$$u = g_1 \quad \text{on } \Gamma_1, \qquad \frac{\partial u}{\partial \boldsymbol{n}} = g_2 \quad \text{on } \Gamma_2$$

に変える。

(b) 自分で選んだ領域を三角形分割して、このプログラムに入力できるデータを生成するプログラムを書く。

7 FreeFem++の文法

7.1 はじめに

すでに FreeFem++ のインストール手順と簡単な解説を行ってある。

FreeFem++ は有限要素法によって微分方程式の数値シミュレーションを行うためのプログラミング言語、そして言語処理系の名前でもある (Hecht [?])。
インタープリターである (その点は MATLAB や Python と似ている)。

今回は、プログラミング言語としての FreeFem++を説明する (マニュアル Hect[?] を見ても良く分からない — 少なくとも私は)。

FreeFem++のことを「有限要素法専用ツール」と考える人もいる。確かに有限要素法に便利な命令が組み込まれているが、それ以外の目的のプログラミングに必要な機能も十分に備わっている (実際、有限体積法や差分法のプログラムも記述可能である)。効率を度外視すれば、C のようなプログラミング言語で出来ることは FreeFem++でも出来る、と考えよう。

文法は、C++に似ている (ゆえに C にも似ている)。C しか知らない人は、C++のストリームを使った入出力 (cout, cin の利用) を調べておくこと。
参考: FreeFem++ は C++ で記述されている。

マニュアル Hect[?] は事例集の性格が強く、言語仕様は整理した形では載っていない。以下の説明は、個人的なノートである桂田 [?] に基づく。

7.1 はじめに　基本的な Poisson 方程式のプログラム

```
// poisson.edp
// 境界の定義（単位円），いわゆる正の向き
border Gamma(t=0,2*pi) { x=cos(t); y=sin(t); }
// 三角形要素分割を生成（境界を 50 に分割）
mesh Th = buildmesh(Gamma(50));
plot(Th,wait=true); // plot(Th,wait=true,ps="Th.eps");
// 有限要素空間は P1（区分的１次多項式）要素
real [int] levels =-0.012:0.001:0.012;
fespace Vh(Th,P1);
Vh u,v;
// Poisson 方程式 -△ u=f の右辺
func f = x*y;
// 問題を解く
solve Poisson(u,v)
 = int2d(Th)(dx(u)*dx(v)+dy(u)*dy(v))-int2d(Th)(f*v)
   +on(Gamma,u=0);
// 可視化（等高線）
plot(u,wait=true);
//plot(u,viso=levels,fill=true,wait=true);
// 可視化（3 次元）--- マウスで使って動かせる
plot(u,dim=3,viso=levels,fill=true,wait=true);
```

→ 独特の命令ばかりで、汎用のプログラミング言語の機能があることは分かりにくい。

7.2 汎用のプログラミング機能

7.2.1 C 言語と良く似ているところ

改めて数えるととても多い。

- // から行末までは注釈、/* と */ で挟まれた部分は注釈 (共に C 言語と同じ)
- 文の最後は ;
- 四則演算 (+, -, *, /) や、代入 (=) などの演算子
- 0 は偽、0 以外の整数は真とみなす。一方、比較演算・論理演算などの結果は 0 (false) または 1 (true).
- 変数宣言の文法も C 言語と同様。型名の後に, で区切った名前のリストを書く。
- 関数呼び出しの文法も C 言語と同様。
- ブロックは { と } で複数 (0 個以上) の文を囲んで作る。
- 比較演算子 (==, !=, <, <=, >, >=)、論理演算子 (&&, ||, !)、`if`, `if else` などの制御構造。
  ただし `switch` はない。
- `for`, `while` などの繰り返し制御。`break` (ループを抜ける), `continue` (次の繰り返し) など。
  ただし `do while` はない。
- 数学関数の名前

他にもあるだろう…

7.2.2 データ型

- 整数を表すための int がある (C 言語の `int` に相当)
- 実数を表すための real がある (C 言語の `double` に相当)
- 複素数を表すための complex がある (C 言語の `complex` に相当, 実部・虚部が `double`)
- 論理を表すための bool がある (C 言語の `bool` に相当). `true`, `false` と言う値があるが、それぞれ 1, 0 の別名と考えて良い。
  例えば `plot(u,wait=true);` は `plot(u,wait=1);` と同じ。
- 文字列を表すための string がある (C++言語の `string` に相当, 日本語不可？)。
  - 2 つの string s1, s2 を、(+ 演算子を用いて) s1+s2 で連結できる。
  - string+数値 とすると、数値を文字列に変換してから連結する。

```
real a=1.23, b=4.56;
string s;
s= "a=" + a + ", b=" + b + ".";
cout << s << endl;
```

  - string を int に変換する `atoi()`, string を real に変換する `atof()` がある (C 言語の真似)。

7.2.3 配列型　1 次元

1 次元配列は、C 言語に (少し) 似ている。

```
real[int] a1(3); // C で double a1[3]; とするのに似ている
for (int i=0;i<3;i++)
  a1[i]=i; // a1(i)=i; としても良い。

real[int] a2 = [0,1,2]; // C で double a[]={0,1,2}; とするのに似てる
real[int] a3 = 0:2; // これは少し MATLAB 風

cout << "a1=" << a1 << endl;
cout << "a2=" << a2 << endl;
cout << "a3=" << a3 << endl;
```

追記: a1 の要素数は a1.n で得られる。

7.2.3 配列型　2 次元

2 次元配列はかなり違う。要素にアクセスするには 名前 (i,j) とする。

```
real[int,int] kuku(9,9);
int i,j;
for (i=0; i<kuku.n; i++) {
  for (j=0; j<kuku.m; j++) {
      kuku(i,j)=(i+1)*(j+1);
      cout << setw(3) << kuku(i,j);
  }
  cout << endl;
}
cout << kuku << endl;

real[int,int] kuku2=[[1,2,3,4,5,6,7,8,9],
                     [2,4,6,8,10,12,14,16,18],
                     [3,6,9,12,15,18,21,24,27],
                     [4,8,12,16,20,24,28,32,36],
                     [5,10,15,20,25,30,35,40,45],
                     [6,12,18,24,30,36,42,48,54],
                     [7,14,21,28,35,42,49,56,63],
                     [8,16,24,32,40,48,56,64,72],
                     [9,18,27,36,45,54,63,72,81]];
cout << kuku2 << endl;
```

参考文献

[1] 菊地文雄：有限要素法概説, サイエンス社 (1980), 新訂版 1999.

[2] 桂田祐史：多変数の微分積分学 2 講義ノート 第 1 部, https://m-katsurada.sakura.ne.jp/lecture/tahensuu2/tahensuu2-p1.pdf (2008).

[3] 桂田祐史：有限要素法への入門, https://m-katsurada.sakura.ne.jp/labo/text/members/fem.pdf. 内容は Poisson 方程式の境界値問題に対する有限要素法です。 これは菊地 [?] の読書ノートの性格が強いので、 一般公開はしていません。 授業を受講する学生に見せることがあります。 (2001〜).

[4] Hecht, F.: New development in FreeFem++, *J. Numer. Math.*, Vol. 20, No. 3-4, pp. 251–265 (2012), https://www.um.es/freefem/ff++/uploads/Main/NewDevelopmentsInFreeFem.pdf.

[5] Hecht, F.: Freefem++, https://doc.freefem.org/pdf/FreeFEM-documentation.pdf, 以前は http://www3.freefem.org/ff++/ftp/freefem++doc.pdf に