FreeFem++数理指向プログラミング

大塚 厚二 *

* 広島国際学院大学 広島市安芸区中野 6-20-1

広島大学 9 月 24 日

- FreeFem++ http://www.freefem.org/ff++/
- FreeFem++日本語 http://comfos.org/jp/ffempp/
- 有限要素法で学ぶ現象と数理—FreeFem++数理思考プログラミング
  http://comfos.org/jp/ffempp/book/

数学をプログラミング言語に

- 十数年前、
  コンピュータがあれば数学はいらない
  「数学よりプログラミングの方が修得が楽」と言われた.
- プログラミングが複雑に，プレハブ工法オブジェクト指向プログラミングが登場
- オブジェクト指向での重要なのは抽象化.
- 数学をプログラミング言語に使えれば良いのでは？
- なぜ駄目なのか？$\Longrightarrow$ 数学を修得するのが難しい
- しかし，プログラミングの最先端は数学以上に難しく

数理指向プログラミング

- プログラミング言語 (数学) の知識は寿命が短い (長い)
  例) アセンブラ (機械優先) → C → Visual C(GUI) → C++
- 数学を学ぶことは，抽象化の訓練に
- ただし，抽象化によって下部構造が完全に隠蔽できたためしがない．数学以外にも知識が必要
  - 流れ制御 (分岐，繰り返し) や入出力
  - 数理情報の可視化
- FreeFem++ vs 数式処理 (Mathematica のような)
  - 矩形領域以外は，解を数式で記述するのは難しい．
  - 数式処理で領域を分割するのは困難
  - もし，数式処理だけで有限要素法を記述すると膨大な計算
- FreeFem++はホワイトボックス型問題解決環境 (PSE)
  - 計算方法を細かく指定できる．
  - 計算 (途中) 結果を抽出できる．

FreeFem++の歴史

1987 MacFem/PCFem：商用ソフト Pascal を使って O.Pironneau が作成

1992 FreeFEM：C++で書き直し．単一メッシュ/P1,P0 要素，アダプティブメッシュ(bamg) O.Pironneau, D.Bernardi, F.Hecht , C.Prudhomme

1996 FreeFem+：複数メッシュ/P1,P0 要素，関数の代数処理，O.Pironneau, D.Bernardi, F.Hecht, K.Ohtsuka

1998 FreeFem++：複数メッシュ/複数有限要素，拡張機能，F.Hecht, O.Pironneau, K.Ohtsuka

1999 FreeFem 3d：S.Del Pino

2008 FreeFem++ V3：1d, 2d, 3d といったマルチ次元での有限要素計算を実現

2014 [http://comfos.org/jp/ffempp/book/] 有限要素法で学ぶ現象と数理ーFreeFem++数理思考プログラミングー(共立出版) 大塚厚二, 高石武史
(以下の p.～は本の場所)

開発メンバー

Frédéric Hecht フランス・パリ第 6 大学
https://www.ljll.math.upmc.fr/J.L.Lions 研究所 (LJLL)
プロジェクトのリーダー，アダプティブメッシュ，有限要素空間など，元 INRIA 研究員

Olivier Pironneau LJLL，INRIA コンサルタント
元リーダー，計算アルゴリズム，流体解析，マニュアルの第 3 章など

Antoine Le Hyaric CNRS, LJLL
並列計算，数値可視化など
統合環境 FreeFem++-cs を開発

Jacques Morice LJLL(ポスドク), 3 次元メッシュと可視化ツール medit の組込

Sylvian Auliac LJLL(大学院生)，非線形最適化ライブラリ (NLopt, IPOPT)，関数最適化計算アルゴリズム (CMA-ES) など

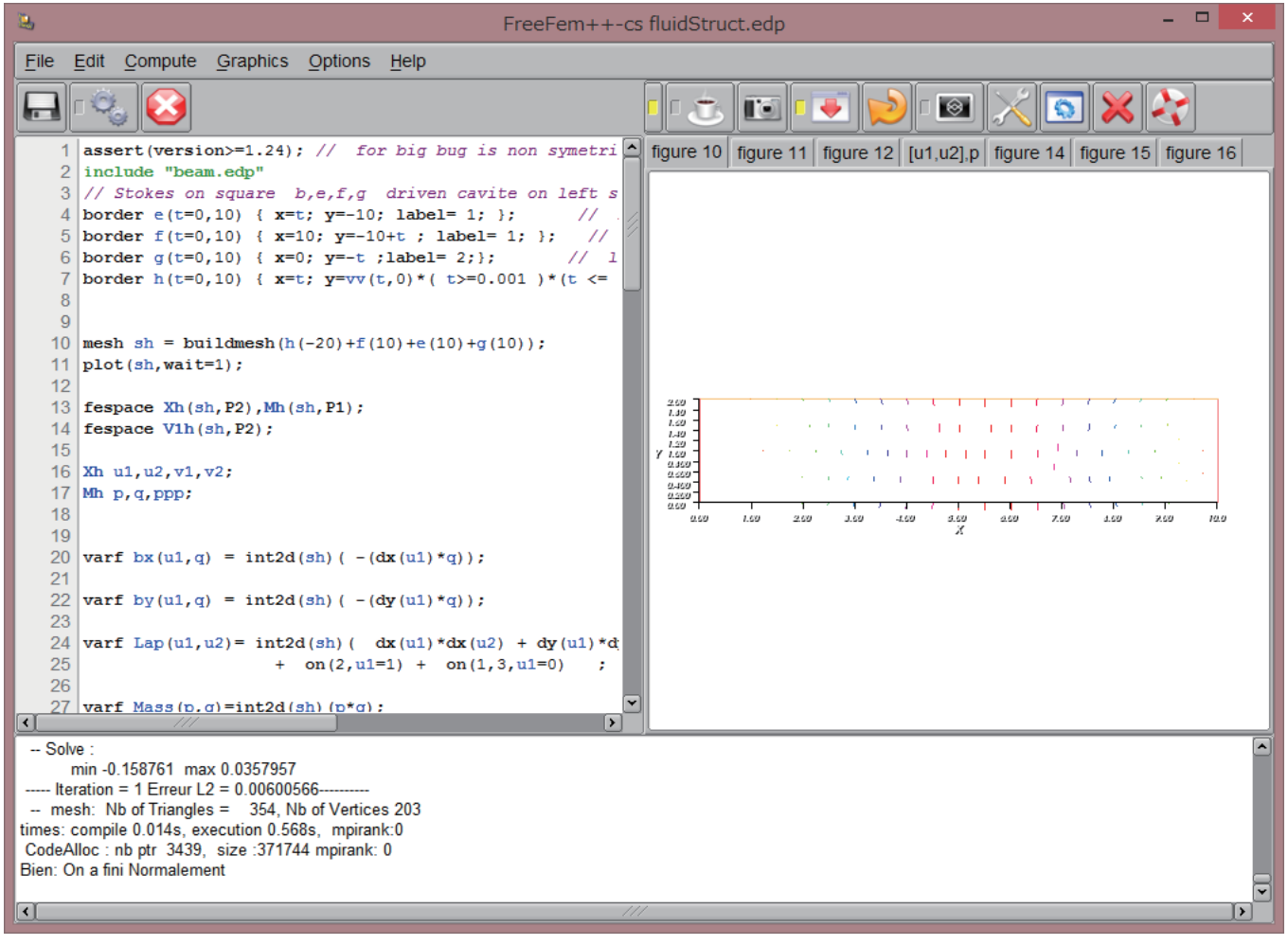
Kohji Ohtsuka 広島国際学院大学
以前，MS-Windows のグラフィック表示など，その

統合開発環境 FreeFem++-cs

開発サイト www.freefem.org/ff++/ に無く，下記から

http://www.ann.jussieu.fr/lehyaric/ffcs/

並列計算などもサポート，日本語は utf-8 のみで直接入力不可



誰のため，目的は

目的は (F.Hecht のスライドから)

1. 研究開発 (R&D)
2. 学術研究
3. 有限要素法，偏微分方程式，弱解や変分形式の教育
4. アルゴリズムの実現性チェックに向けた試作
5. 数値実験
6. 計算科学や並列計算 (MPI)

誰のため：研究者，エンジニア，大学教員，学生…

メーリングリスト：Freefempp@ljll.math.upmc.fr
一日に 5〜20 通，377 人のメンバー

1000 人以上の実利用者 (月に 200 以上のダウンロード)

主な機能 (2d,3d) I

1. 多くの有限要素：連続 P1,P2 要素，不連続 P0, P1, RT0,RT1,BDM1-要素，辺要素，MINI 要素など
2. メッシュ非依存性：メッシュ$\mathcal{T}^1$ で計算した有限要素関数 f を他のメッシュ$\mathcal{T}^2$ の有限要素関数や配列に補間する.
3. ベクトルや行列を使った弱形式による複素・実問題の定義
   有限要素法のアルゴリズムの多くは，行列やベクトルで記述される.
4. 不連続 Galerkin による定式化 (現在 (2010) は二次元のみ)
5. 複数の行列解法： LU, Cholesky, Crout, CG, GMRES, UMFPack, SuperLU, MUMPS, HIPS , SUPERLU_DIST, PASTIX. ... sparse linear
6. 固有値問題：ARPACK を使った解法
7. 統合環境：FreeFem++-cs
8. 境界の解析的表現

主な機能 (2d,3d) II

9. 自動メッシュ分割：Delaunay-Voronoï分割 (2d,3d(tetgen))
10. メッシュや有限要素解の保存・読み込み
11. アダプティブメッシュ： 解などの Hessian を使った距離に基づく
12. 動的リンク機能：parview, gmsh , vtk, medit, gnuplot といった外部のプログラムを追加することで機能を拡張できるような機構
13. 並列処理の MPI をフルにサポート
14. 非線形最適化ツール：CG, Ipopt, NLOpt, stochastic
15. 豊富なサンプル：Navier-Stokes (2d/3d), elasticity (2d/3d), 流体・構造連成問題, 固有値問題, Schwarz の領域分割法, 残差誤差指標

「有限要素法で学ぶ現象と数理」には，形状最適化設計や反応拡散問題の計算 (高石) がある．

有限要素法とは

有限要素法は偏微分方程式境界値問題の数値解法で，変分法に基づため数学理論が明確である．次の５つのステップから構成される．

1. メッシュ$\mathcal{T}_h(\Omega)$ 自動生成　三角形分割 (2d)，四面体分割 (3d)
2. 有限要素空間 $V_h(\mathcal{T}_h(\Omega), \bullet)$ を生成
3. 弱形式から連立方程式の生成

$$\text{強形式}\quad -\Delta u = f \quad (\Omega\text{上}), u = 0 \quad (\partial\Omega\text{上})$$

$$\text{弱形式}\quad a(u,v) - \ell(v) = 0, \quad \ell(v) = \int_\Omega f v \, dx$$

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \qquad \text{だたし } u = 0(\partial\Omega\text{上})$$

4. 連立方程式 $A\boldsymbol{u} = \boldsymbol{F}$ を解く　$A = \Big(a(\phi_j, \phi_i)\Big), \boldsymbol{F} = \Big(\ell(\phi_i)\Big)$
5. 解の評価 (数値可視化, 解の改良など)

ex01.edp (p.28) I

```
1:  border C(t=0,2*pi){x=cos(t); y=sin(t);} //単位円周
2:  mesh Th = buildmesh (C(50)); //1. メッシュ生成 T_h
                                          //(図 3 参照)
3:  fespace Vh(Th,P1);  //2. 有限要素空間 V_h P1 要素
4:  Vh u,v;                                //u,v∈V_h
5:  func f = -1;          //既知関数 f(x,y) = −1
6:  func g = 0;           //g(x,y) = 0
7:  problem Poisson(u,v) = //3. 弱形式 弱形式 (2)
8:     int2d(Th)(dx(u)*dx(v)+dy(u)*dy(v))
                  //a(u,v) = ∫_Ω ∇u·∇v
9:     - int2d(Th)(f*v)       //−ℓ(v) = ∫_Ω fv
10:          + on(C,u=g);     //u = g (∂Ω 上)
11:  Poisson;      //4. 数値解を求める
12:  plot(u);     //5. 解の等高線表示 図 2 参照
```

ex01.edp (p.28) II

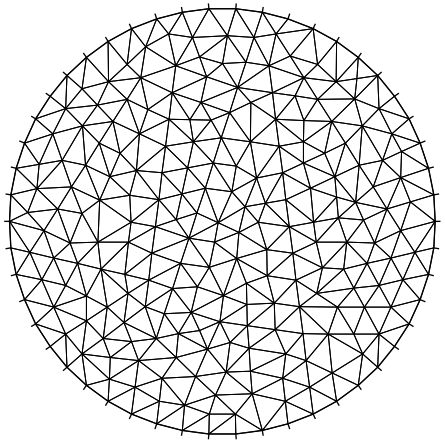


図: 三角形分割 Th(= $\mathcal{T}_h(\Omega)$)

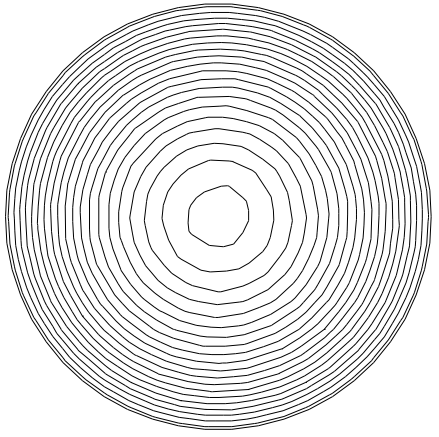


図: plot(u) 等高線表示

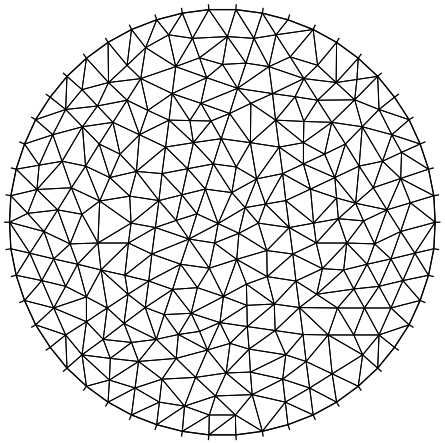
2.2 メッシュ(p.32), 3.1 メッシュ分割 (p.69)

FreeFem++では三角形分割，四面体分割といったメッシュ生成について複数の方法を提供している．

1. 標準的な方法では，領域の境界 $\partial\Omega$ を複数の曲線 $\Gamma_i, i = 1, \cdots, L$ で構成し，Γ_i を次のようにパラメータ表示する．
$$\Gamma_i = \{(\varphi_x(t), \varphi_y(t)) : \ \alpha \le t \le \beta\}$$
2. 縦横の分割数を指定して長方形を分割する (square)．有限要素法のテスト，アルゴリズムの検証，そして計算精度をチェックする場合には便利である．
3. CAD ソフトなどで生成したメッシュファイル xxxxx.msh を
```
mesh Th = readmesh("xxxxx.msh");
```
で読み込む．以後，メッシュは Th を参照するだけで利用できる．(3.1,p.69)
4. 画像の境界を認識してメッシュを生成する．(3.1,p.72)

1.1 境界を定義

```
1:  border C(t=0,2*pi){x=cos(t); y=sin(t);} //単位円周
2:  mesh Th = buildmesh (C(50)); //1. メッシュ生成 $\mathcal{T}_h$
```



領域 Ω を決め，メッシュ分割 $\mathcal{T}_h(\Omega)$ を生成．使われる予約語は border, mesh, buildmesh.

図: 2: 三角形分割 Th(= $\mathcal{T}_h(\Omega)$)

smileface.edp(p.37) I

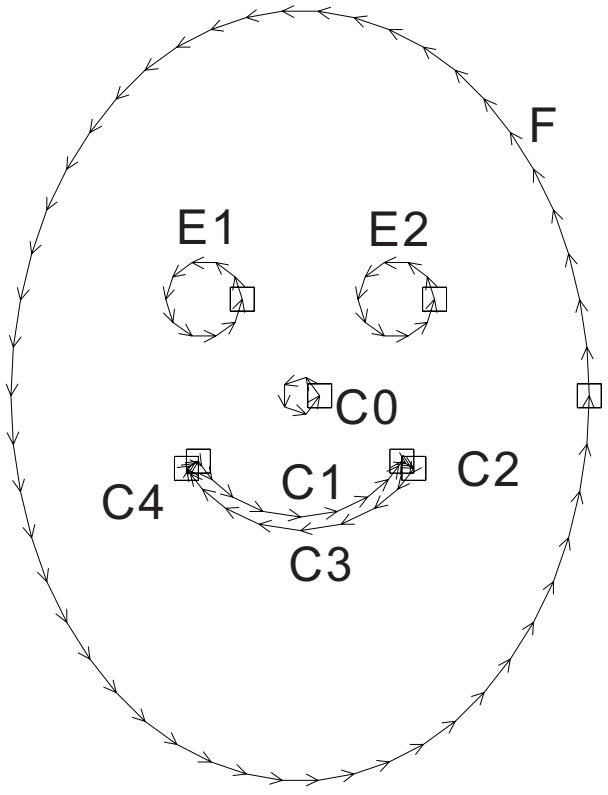


図: 笑う顔 (smile face) 曲線

三角形分割の例) 図 4 では, F,E1,E2,C0,C1,C2,C3,C4 と 8 曲線が使われている.

3 次元領域 Ω の境界 $\cup\Gamma_j$(曲面) は movemesh23 で生成し, 内部への 4 面体分割は tetg() で生成 (p.83)

smileface.edp(p.37) II

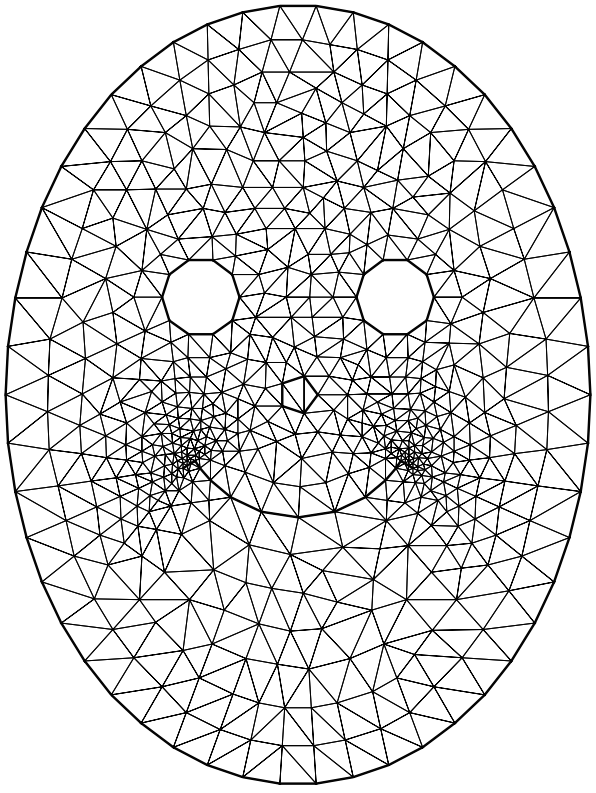


図: 笑う顔の三角形分割

図５のような領域になるには曲線の向きが重要になる．境界には buildmesh で指定した順番で，F(1)，E1(2)，E2(3)，C1(4)，C2(5)，C3(6)，C4(7)，C0(8) と番号が付く．

3 次元メッシュ(境界＝曲面の和)p.83

3 次元領域 Ω を境界面

$$\Gamma = \cup_{j=1}^{J} \Gamma_j \quad \text{(曲面の和)}, \quad \Gamma_j = \Phi^j(\Pi_j)$$

ここで，Π_j は平面，$\Phi^j(x, y)$ は写像.

mesh3 $\mathcal{T}_h(\Gamma_j)$=movemesh23($\mathcal{T}_h(\Pi_j)$,transfo=[$\Phi_1^j, \Phi_2^j, \Phi_3^j$]);

として 3 次元曲面のメッシュ$\mathcal{T}_h(\Gamma_j)$ を作り，

mesh3 $\mathcal{T}_h(\Gamma)$=$\mathcal{T}_h(\Gamma_1) + \cdots + \mathcal{T}_h(\Gamma_J)$;

で境界面の三角形分割が出来る．ただし，メッシュ曲面 $\mathcal{T}_h(\Gamma_j)$ は Γ_j とはずれるため $\cup_{j=1}^{J} \Gamma_j$ と一致しないだけでなく，閉曲面にならない場合もある．曲面から内部への分割は Tetgen を使って次のように内部に切っていく.

mesh3 Th=tetg($\mathcal{T}_h(\Gamma)$,switch="paAAQy",regionlist=domain);

3 次元メッシュ(Buildlayers) p.84

平面領域 Π で定義された関数 $z_l(x,y) < z_u(x,y)$ があるとき，空間

$$\Omega = \{(x,y,z) : (x,y) \in \Pi, z_l(x,y) < z < z_u(x,y)\}$$

を 4 面体分割する命令として buildlayers()(p.84) があり，回転体生成にも使える.

mesh3 $\mathcal{T}_h(\Omega)$=buildlayers($\mathcal{T}_h(\Pi)$, nz, zbound=[z_l, z_u]);

ここで nz は z 軸方向の分割数 (層の数).
z 軸方向をパラメータにして平面 Π の回転体を次のように作ることができる.

mesh3 $\mathcal{T}_h(\Omega)$=buildlayers($\mathcal{T}_h(\Pi)$, nz, zbound=[0,$e\pi$],
transfo=[$y\cos(z), y\sin(z), x$]);

1.2 長方形 (square) p.33 I

正方形領域 $\Omega = (0,1)^2$ を横４分割，縦５分割する場合は

```
mesh Th = square(4,5);
```

とする．長方形領域 $\Omega = (x_0, x_1) \times (y_0, y_1)$ を $n \times m$ に分割する場合は，

```
mesh  Th = square(n,m,[x0+(x1-x0)*x,y0+(y1-y0)*y]);
```

とする．たとえばプログラム

```
real x0=0.1,x1=1.5;
real y0=0.2,y1=0.8;
int n=5,m=10;
mesh Th=square(n,m,[x0+(x1-x0)*x,y0+(y1-y0)*y]);
```

による四角形は図６のようになり，境界 (辺) には番号が付く．

1.2 長方形 (square) p.33 II

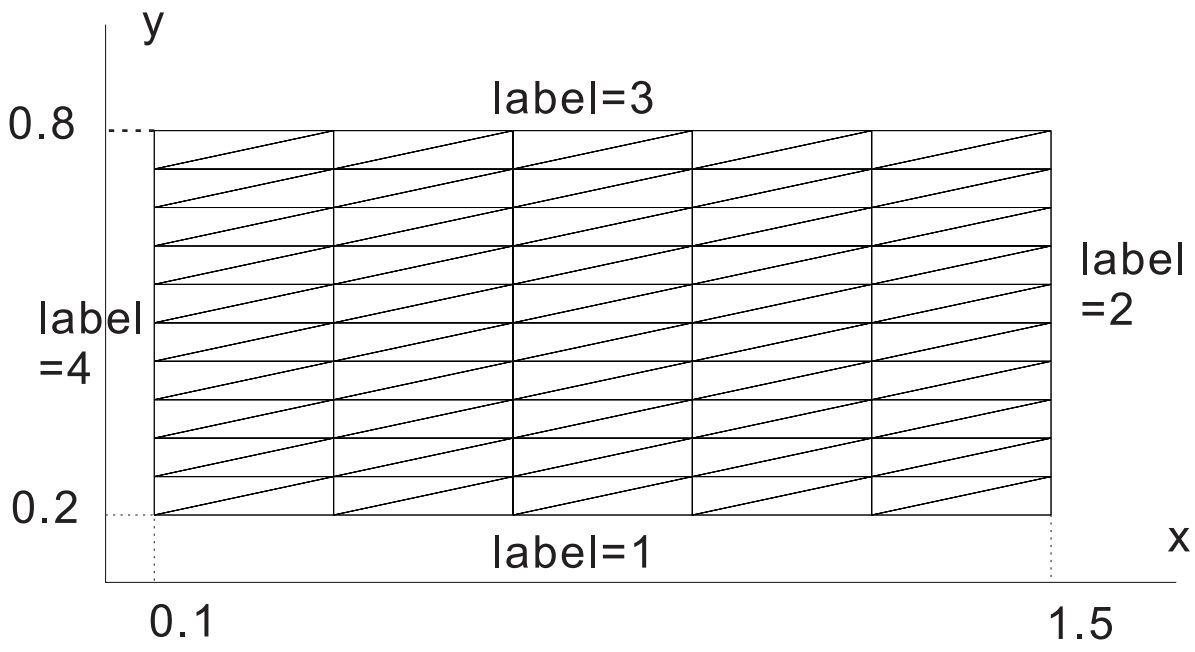


図: square による三角形分割

1.3 メッシュデータファイル

生成したメッシュを他のプログラムで再利用したい，CAD などで生成したメッシュを FreeFem++で利用したい.

savemesh 生成したメッシュを再利用するために外部記憶装置に保存 (p.70)

readmesh データ形式 (BAMG, 表 3.1(p.89)) のファイルを読み込んでメッシュを生成する.

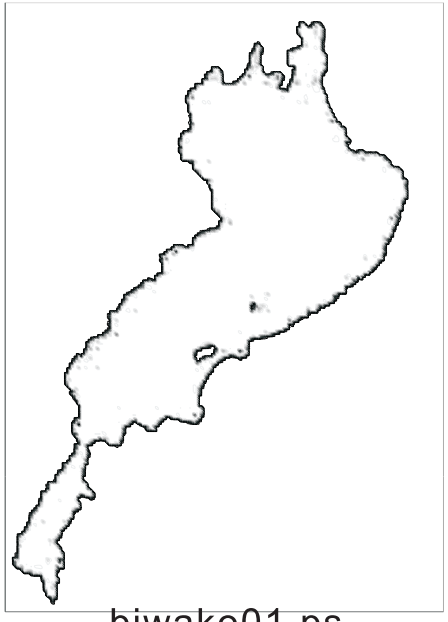
```
mesh Th=readmesh("xxxxx.mesh"); //２次元
mesh3 Th=readmesh3("xxxxx.mesh"); //３次元
```

２次元の場合は，より単純なデータ形式 (msh, 図 3.1(p.71)) がある.

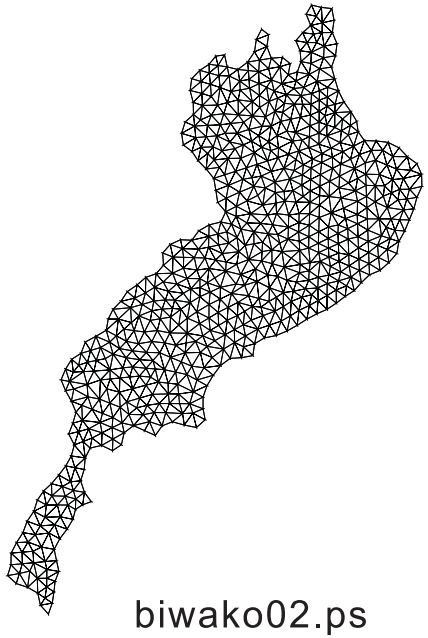
```
mesh Th=readmesh("xxxxx.msh");
```

1.4 画像データからの生成

biwako.pgm



biwako01.ps



biwako02.ps

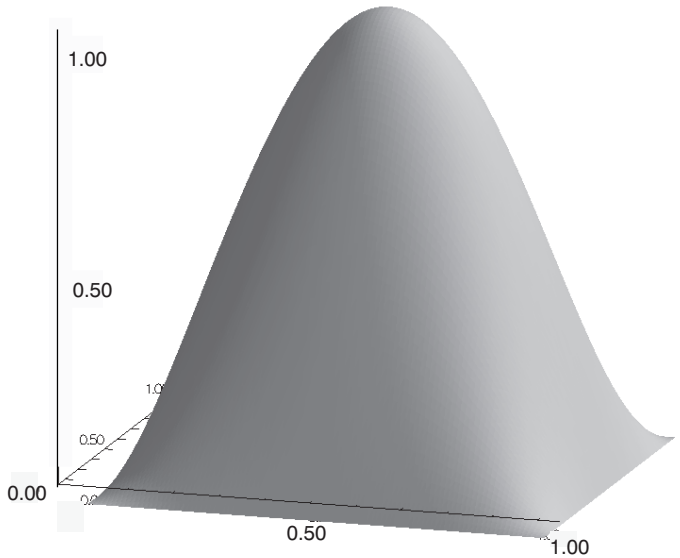
図: biwako2.edp(p.72) 元画像 (3 行目)，縁を抽出した画像 (15 行目)，生成されたメッシュ(30 行目)

2. 有限要素空間を生成 I

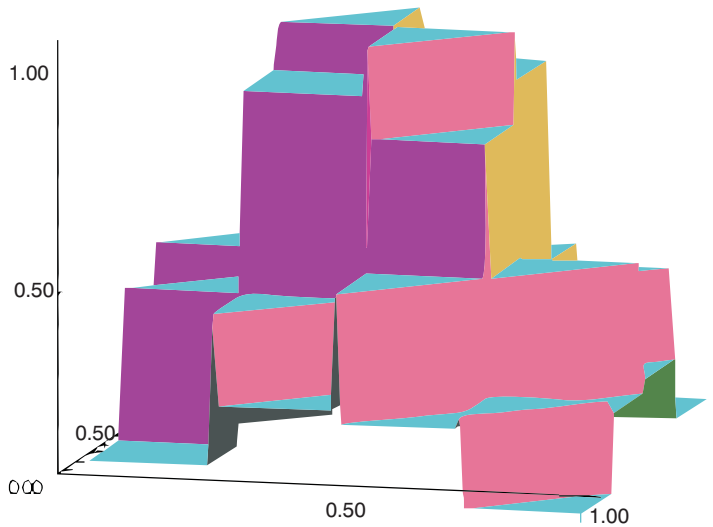
fespace $V_h(\mathcal{T}_h, \bullet) = \left\{ v : v = v_1\phi_1 + \cdots + v_M\phi_M,\ v_i \in \mathbb{R} \right\}$

型の指定 $\bullet$ は基底関数を選択することを表す.

```
Vh f=cos(pi(x-0.5))*sin(pi(y-0.5)); //Vh =V_h( T_h,●)
```

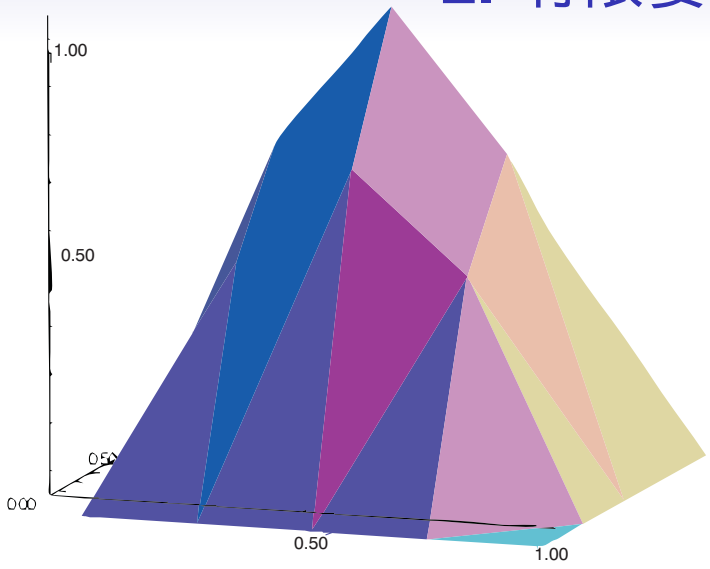


$f(x,y) = \cos(\pi(x - \frac{1}{2}))\cos(\pi(y - \frac{1}{2}))$

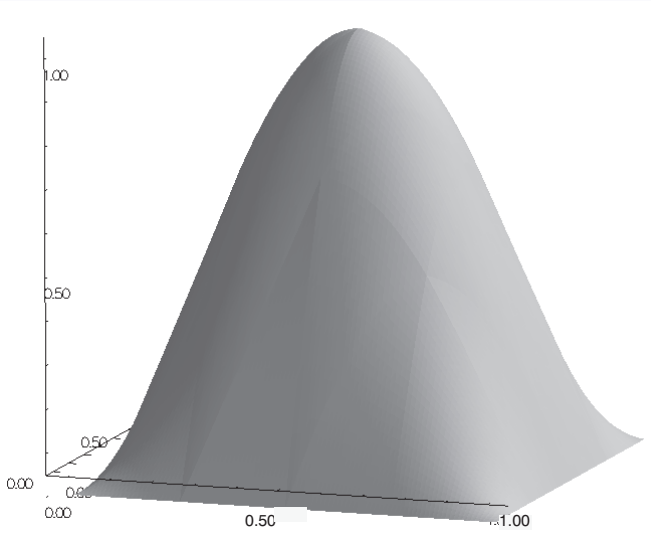


$\bullet$ = P0

2. 有限要素空間を生成 II



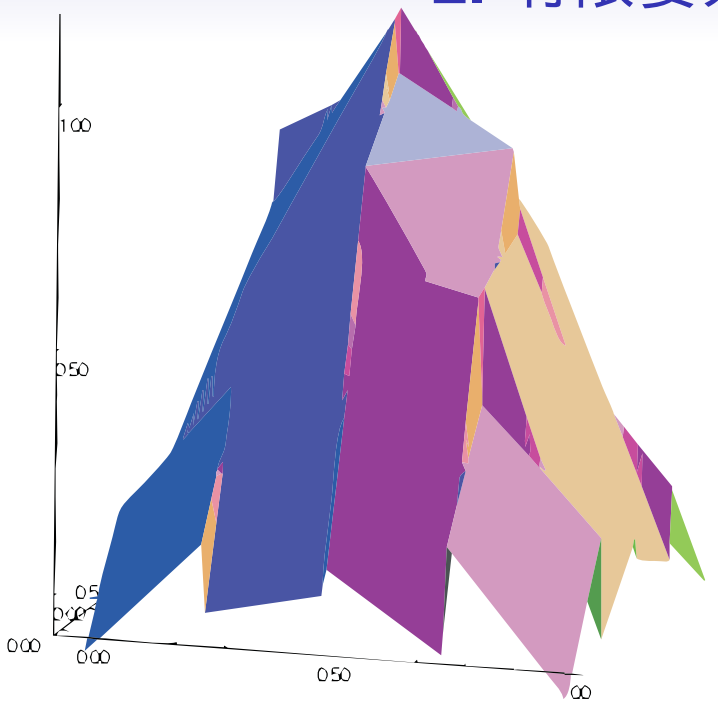
- =P1

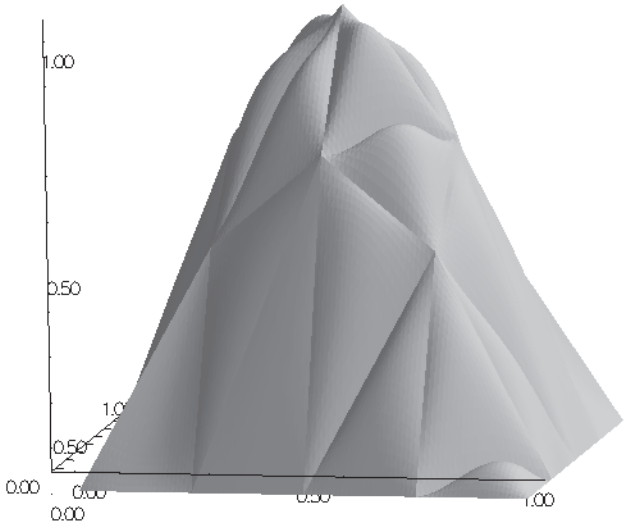


- =P2

図: $f(x,y) = \cos(\pi(x-\frac{1}{2}))\cos(\pi(y-\frac{1}{2}))$ のグラフと有限要素空間への射影 (補間)

2. 有限要素空間を生成 III





- =P1nc 1 次非適合要素
- =P1b 流体：バブル要素,MINI 要素

図: $f(x,y) = \cos(\pi(x-\frac{1}{2}))\cos(\pi(y-\frac{1}{2}))$

2. 有限要素空間を生成 IV

P0,P1,P2,P1b 3 次元問題 (P03d,P13d,P23d,P1b3d) も O.K. (p.89)

P3 2 次元問題のみ O.K.?

Raviart-Thomas 要素 三角形要素の各辺を通過する流束を使うベクトル値関数, 3 次元問題も O.K. (● =RT03d) (p.92) (p.92)

P1dc, P2dc 1 次・2 次不連続要素 (マニュアル)

Edge03d ベクトル値関数に対する 0 次エッジ要素 (マニュアル)

3. 弱形式から連立方程式の生成 I

2 階偏微分方程式境界値問題を FreeFem++は，$H^1(\Omega)$ でしか解けない．そのため Dirichlet 境界条件をペナルティ法で解く．

$$-\Delta u = f \quad \Omega\text{内}; \qquad u = g \quad \Gamma\text{上}\ (\Gamma = \partial\Omega) \tag{1}$$

FreeFem++では，強形式 (1) をペナルティ法を使って双線形 $a(\cdot,\cdot)$ 及び線形 $\ell(\cdot)$ とに分け，

$$a(u,v) = \ell_g(v) \qquad \forall v \in V_h$$

といった弱形式で解く．P1 要素のときは

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \, d\Omega + \mathrm{on}(\Gamma, u = 0); \tag{2}$$

$$\ell_g(v) = \int_\Omega f v \, d\Omega + \{g_i^\epsilon\} \tag{3}$$

$$g_i^\epsilon = \epsilon^{-1} g(q^i) \quad q^i \in \Gamma; \text{それ以外 } g_i^\epsilon = 0 \tag{4}$$

3. 弱形式から連立方程式の生成 II

連立方程式 $A\boldsymbol{u} = \boldsymbol{F}$ を解く

$$\mathtt{A} \quad = \quad (a_{ij}) = \int_\Omega \nabla\phi_j \cdot \nabla\phi_i \, d\Omega \quad (i \neq j); \tag{5}$$

$$a_{ii} \quad = \quad \epsilon^{-1} \quad (\text{節点 } q^i \in \partial\Omega \text{のとき}), \quad a_{ii} = \int_\Omega \nabla\phi_i \cdot \nabla\phi_i \, d\Omega \quad q^i \notin \partial\Omega$$

$$\boldsymbol{F} = \{F_i\}, \quad F_i = \ell(\phi_i) = \int_\Omega f\phi_i \, d\Omega \left(+\epsilon^{-1} g(q^i) \quad q^i \in \partial\Omega \text{のとき} \right)$$

【注意】剛性行列 A を生成するとき，数値計算法により要素の格納法 (圧縮) が異なるため，`solver=`を使って数値計算法を指定できる．指定しない場合は，バージョンごとに異なる数値計算法
`LU,Crout,Cholesky,UMFPACK,CG,GMRES`
が使われる (p.50).

弱形式による記述 I

境界を $\Gamma = \cup\Gamma_j$ としてで $u = 0$ $(\Gamma_i$上) なら，境界値問題の名前を Prob として

```
func f=f(x,y);
func g=g(x,y);
Vh u,v: //u,v∈ Vh(Ω) とする.
problem Prob(u,v)=
        a(u,v)
        //例えば,∫Ω ∇u∇v=int2d(Th)(dx(u)*dx(v)+dy(u)*dy(v)
         -ℓ(v) //例えば, ℓ(v)=int1d(f*v)
         +on(i,u=g);
```

連立方程式の解法を • に選択する場合は,

```
problem Prob(u,v,solver=•)=
         a(u,v)
```

4. 連立方程式を解く

3. 弱形式から連立方程式の生成における連立方程式

$$Au = F$$

は弱形式の名前を書くだけ.

```
Prob;
plot(u); //数値解 u= u_h の等高線を表示
```

弱形式を書いて，同時に連立方程式を解く次の命令がある.

```
solve Prob(u,v,solver=●)=
        a(u,v)
```

algebra.edp (p.109)

```
1:  border C(t=0,2*pi){x=cos(t); y=sin(t);}
2:  mesh Th = buildmesh (C(10));
3:  fespace Vh(Th,P1);
4:  Vh u,v,F;                    //解 u, テスト関数 v
5:  func f= -1000; 6:  func g=10;
7:    //Step1, 3 での u,v はダミー
8:  varf a(u,v) = int2d(Th)( dx(u)*dx(v) + dy(u)*dy(v))
     + on(C,u=0) ;
9:  matrix A=a(Vh,Vh);  //Step2
10:  varf b(u,v) = int2d(Th)( f*v )+on(C,u=0 ); // Step3
11:  F[] = b(0,Vh);  //Step4
12:  matrix B=b(Vh,Vh);  //B = (b_ij), b_ii = ε^-1  q^i∈Γ 以外 0
13:  Vh gh = g;
14:  F[] += B*gh[];  //+ε^-1 g(q^i), q^i∈Γ
15:  u[]=A^-1*F[];   //連立方程式 Au[] = F[] を解く
```

メッシュ非依存性 I

数値計算結果 u= u_h はメッシュ Th= $\mathcal{T}_h(\Omega)$ 上の有限要素関数
$u_h \in V_h(\mathcal{T}_h(\Omega), \bullet)$
ua$\in aV_h(\mathcal{T}_h^a(\Omega), \bullet)$ such that ua(q^i)=u(q^i), $q^i \in \Omega$(節点) を生成

```
aVh ua = u;   //異なるメッシュへの補間
```

【応用例】
1. 誤差評価　異なるメッシュサイズ $h < h'$ での $u_h \in V_h(\mathcal{T}_h(\Omega), \bullet)$, $u_{h'} \in V_{h'}(\mathcal{T}_{h'}(\Omega), \bullet)$ を比較 (precision02.edp, p.99)

$$\|u_h - u_{h'}\|_{L^2(\Omega)}$$

メッシュ非依存性 II

2. 領域分割法 schwarz-overlap.edp (p.102)

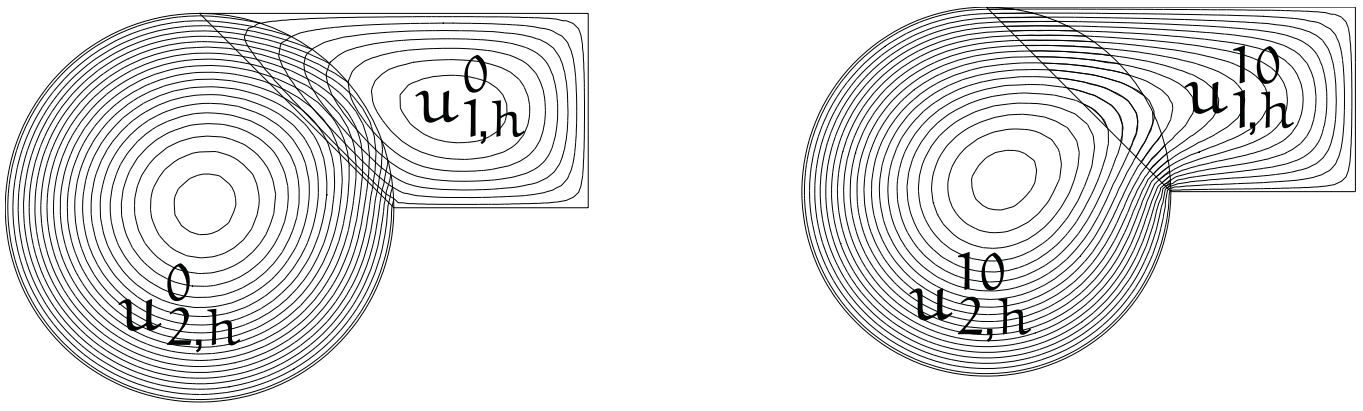


図: schwarz の overlap 法

メッシュ非依存性 III

3. ズーミング airfoil.edp (p.106)

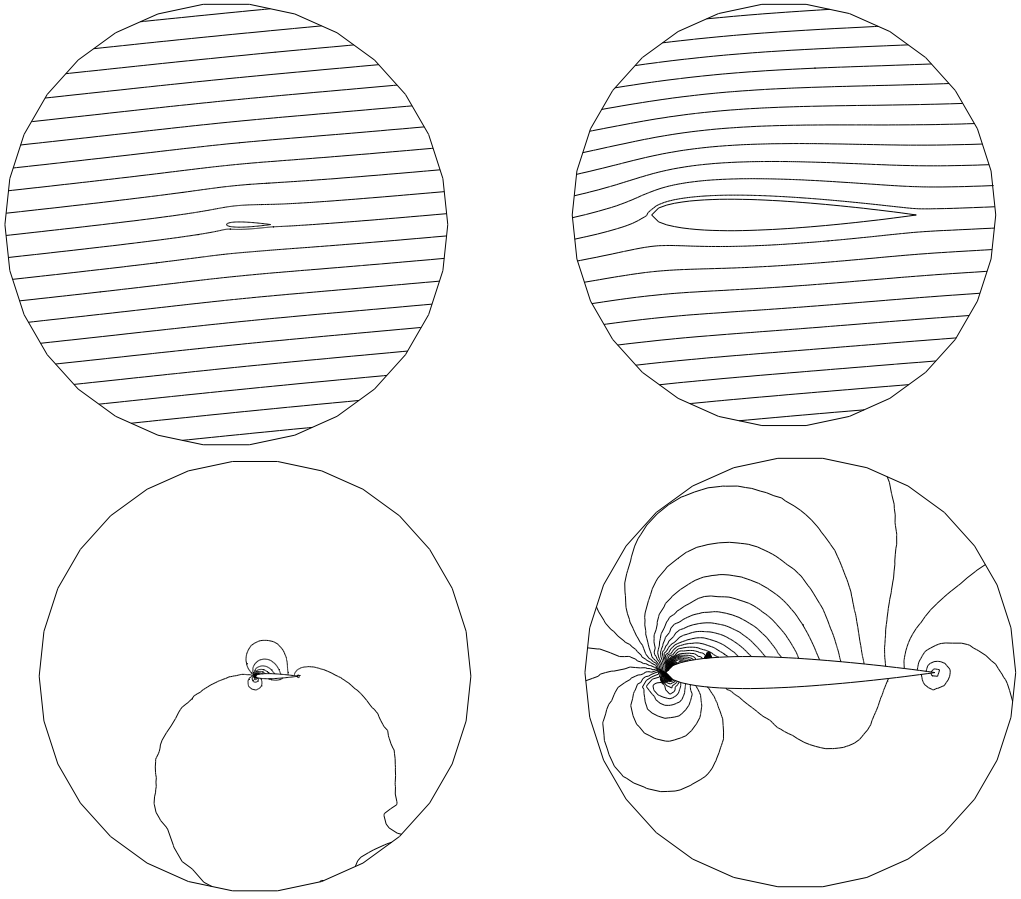


図: 左側が全域で右側が翼周り．上から翼周りの流れ，下が揚力の等高線

5. 解を評価する I

数値計算可視化: 膨大な数となる数値計算結果をコンピュータグラフィックで表示してデータに含まれる科学的な特徴や意味を直感的に理解する.

```
plot(a(10)+b(10)+c(10));  //曲線 a∪b∪c を表示
plot(Th);  //メッシュTh を表示
plot(u);  //有限要素関数 u を表示
plot([u,v]);  //ベクトル値 [u,v]= (u,v) のベクトル表示
```

データ抽出: FreeFem++にはホワイトボックス・テストができる各種操作がある.

1. 付録 A.9(メッシュ分割での操作,p.227) メッシュ要素 (P0 要素), 節点, サイズなどを取り出せる.
2. 付録 A.10(有限要素空間での操作,p.228) 有限要素関数の基底数 (自由度), 第 k 要素の i 番目節点番号など

5. 解を評価する II

数学評価: 厳密解と近似解とを L^2 ノルムなどで評価する

$$\|u - u_h\|_{L^2(\Omega)} \leq Ch^2 \qquad \text{(p.208)}$$

における係数 C を推定する (precision02.edp, p.99).

統計処理: 計算結果をファイルに保存し，グラフにするなど統計処理を行う.

1. 冷却の問題での温度変化 (heatS.edp, p.64)
2. 熱方程式を空間有限要素・時間差分 θ 法で解くときの誤差評価 (evolution.edp,p.214)
   $\theta = 0$ 前進差分
   $\theta = \frac{1}{2}$ Crank-Nicolson
   $\theta = 1$ 後退差分

5. 解を評価する III

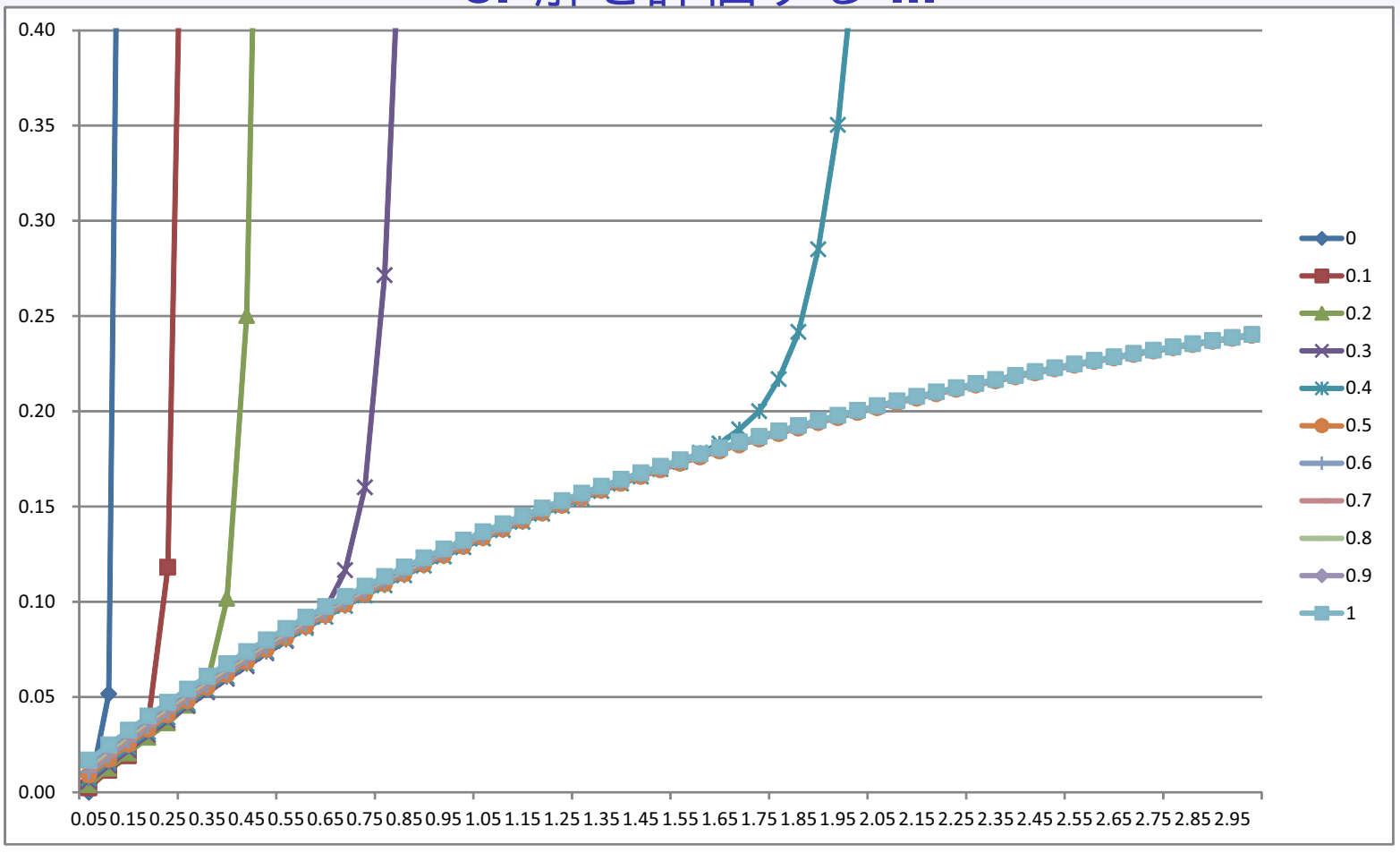


図: $n = 0, 1, \cdots$ での相対誤差 $\|u_h^n(\theta) - u_{ex}(n\tau)\|_{L^2(\Omega)} / \|u_{ex}(n\tau)\|_{L^2(\Omega)}$ をプロット

5. 解を評価する IV

評価用プログラム: メッシュサイズを変えてみたり，アダプティブメッシュを使うなど試行して，可視化・データ抽出などにより最適なメッシュを検討する.

【例】L 字領域で $-\Delta u = -1, \quad u = 0$ ($\partial\Omega$上)(p.36)

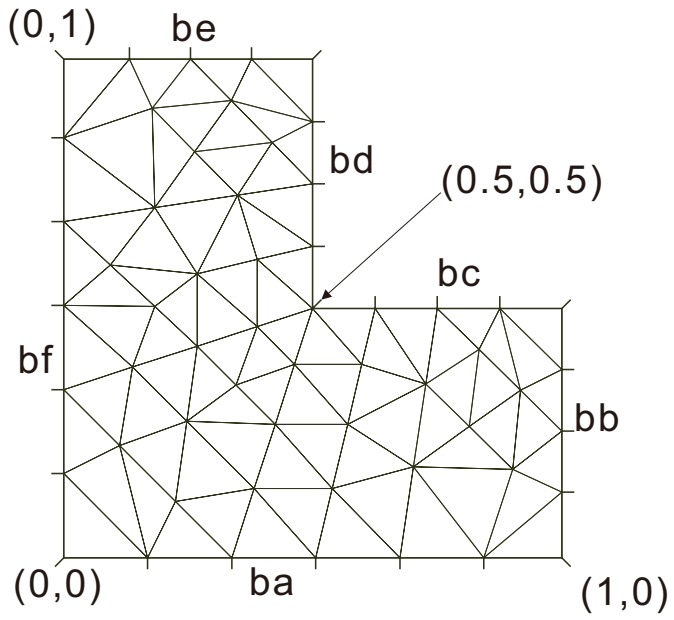


図: 三角形分割 Th

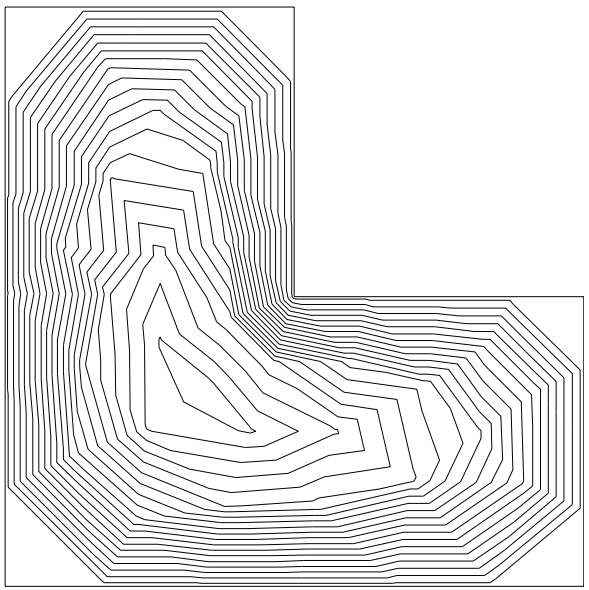
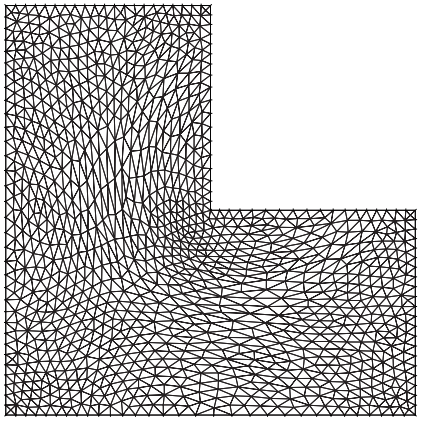


図: plot(u) による解の等高線表示

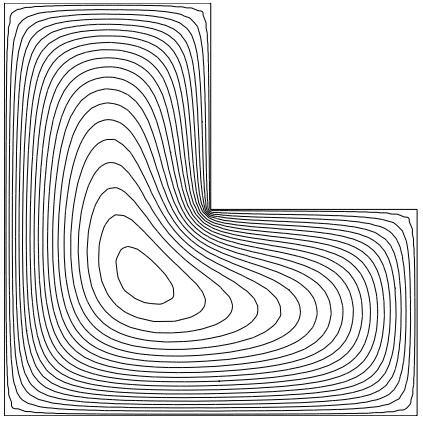
5. 解を評価する V

```
Th = adaptmesh(Th,u); (p.61)
```

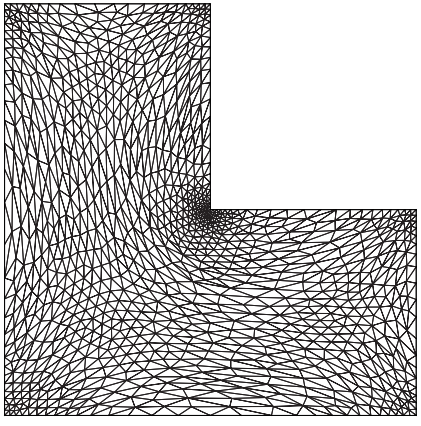
$i = 1$



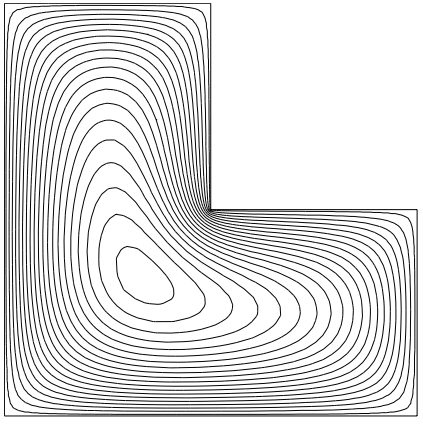
plot(u)



$i = 2$



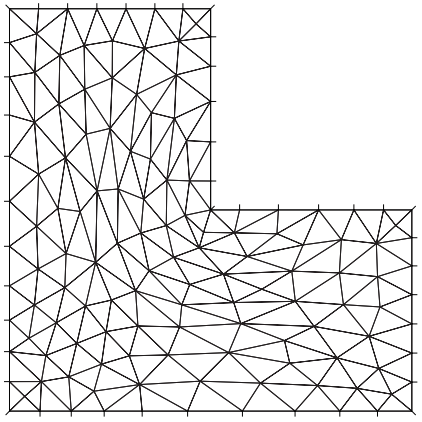
plot(u)



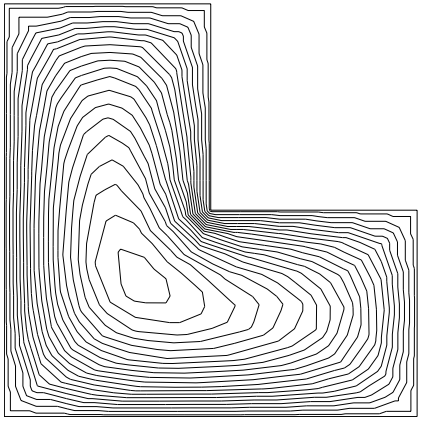
5. 解を評価する VI

```
Th = adaptmesh(Th,u,err=•); (p.62)
```

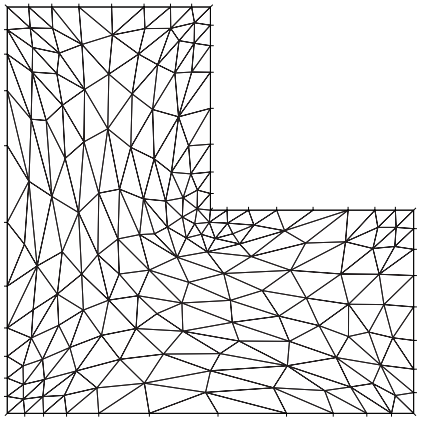
$i = 1$



err=0.1



$i = 2$



err=0.06

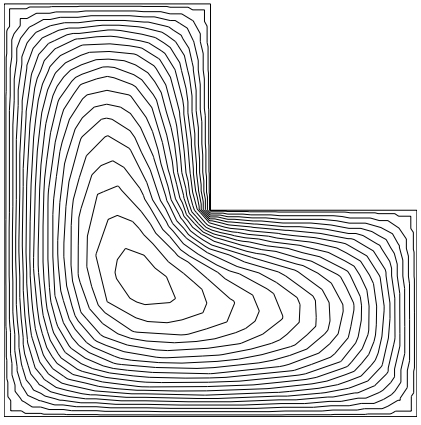
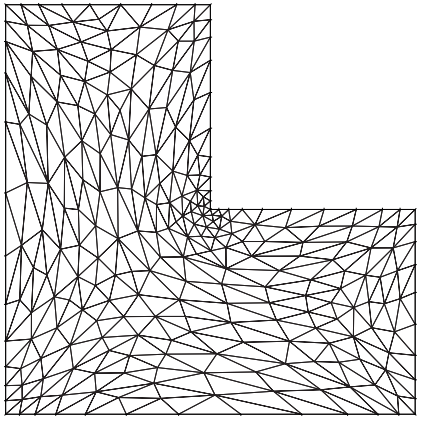


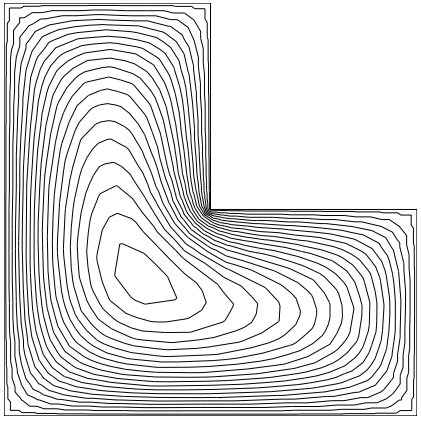
図: オプション err=を制御したアダプティブメッシュ

5. 解を評価する VII

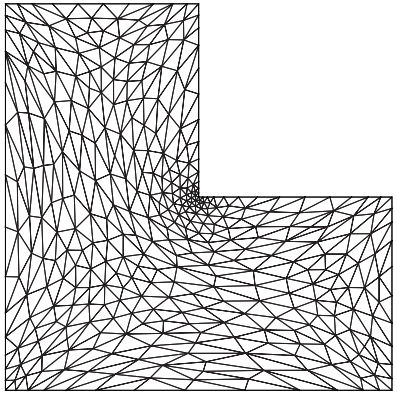
$i = 3$



err=0.04



$i = 4$



err=0.03

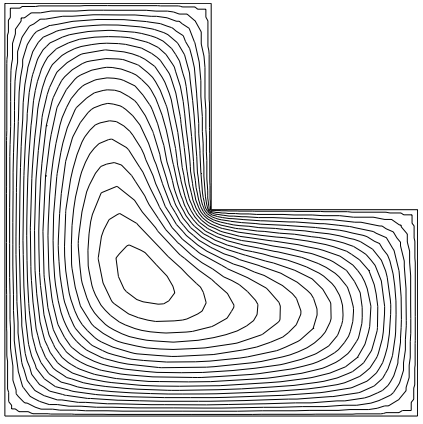


図: オプション err=を制御したアダプティブメッシュ

5. 解を評価する VIII

FreeFem++では行列 $\mathcal{M}$ を使った 2 点 P,Q 間の距離

$$\| \overrightarrow{\mathrm{PQ}} \|_{\mathcal{M}} = \sqrt{\langle \overrightarrow{\mathrm{PQ}}, \mathcal{M} \overrightarrow{\mathrm{PQ}} \rangle} \qquad \mathcal{M} = \begin{pmatrix} m_{11} & m_{12} \\ m_{12} & m_{22} \end{pmatrix}$$

の下に Delaunay-Voronoi アルゴリズムを適用することでアダプティブメッシュ

```
Th = adaptmesh(Th, f, オプション);
```

を実現している.

$$\mathcal{M} = \frac{1}{\epsilon}|\mathcal{H}|, \quad \mathcal{H} = \nabla\nabla f = \begin{pmatrix} \partial^2 f/\partial x^2 & \partial^2 f/\partial x \partial y \\ \partial^2 f/\partial x \partial y & \partial^2 f/\partial y^2 \end{pmatrix} \tag{6}$$

- 通常は $f = \mathrm{u}$ (有限要素解)
  解の漸近挙動 $u \simeq u_S$ を使い $f = u_S$(p.79)
- オプション err=ϵ (p61)