FreeFem++, part IV

M. M. Sussman

sussmanm@math.pitt.edu

Office Hours: 11:10AM-12:10PM, Thack 622

May 12 – June 19, 2014

Topics

3D

- TetGen
- Layer meshes

Mesh adaptation

Topics

3D

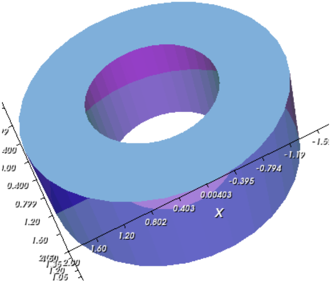
- TetGen
- Layer meshes

Mesh adaptation

Using TetGen inside FreeFem++

1. Generate 2D meshes for each face
2. Type **`mesh`**
3. **`movemesh23`**
   - Place the face meshes in (x, y, z)-space
   - Transform the faces meshes
   - Type **`mesh3`**
4. "Glue" the faces together
5. Use **`tetg`** to get volumn mesh
6. Type **`mesh3`**
7. Use **`movemesh3`** to transform the mesh

Example 40: mesh on a “washer”



1. Generate a mesh on a rectangular parallelopiped
2. Map it to a thick cylinder (washer)

example40.edp code

```
verbosity=2;
load "msh3"
load "tetgen"
load "medit"

// front (z=0) and back (z=1.5) faces
real x0, x1, y0, y1;
x0=1.;
x1=2.;
y0=0.;
y1=2*pi;
mesh Thsq1 = square(5, 35, [x0 + (x1-x0)*x, y0 + (y1-y0)*y] );
```

example40.edp code

```
verbosity=2;
load "msh3"
load "tetgen"
load "medit"

// front (z=0) and back (z=1.5) faces
real x0, x1, y0, y1;
x0=1.;
x1=2.;
y0=0.;
y1=2*pi;
mesh Thsq1 = square(5, 35, [x0 + (x1-x0)*x, y0 + (y1-y0)*y] );

func ZZ1min = 0;
func ZZ1max = 1.5;
func XX1 = x;
func YY1 = y;

int[int] ref31h = [0, 12];
int[int] ref31b = [0, 11];
```

example40.edp code

```
verbosity=2;
load "msh3"
load "tetgen"
load "medit"

// front (z=0) and back (z=1.5) faces
real x0, x1, y0, y1;
x0=1.;
x1=2.;
y0=0.;
y1=2*pi;
mesh Thsq1 = square(5, 35, [x0 + (x1-x0)*x, y0 + (y1-y0)*y] );

func ZZ1min = 0;
func ZZ1max = 1.5;
func XX1 = x;
func YY1 = y;

int[int] ref31h = [0, 12];
int[int] ref31b = [0, 11];

mesh3 Th31h = movemesh23(Thsq1, transfo=[XX1, YY1, ZZ1max],
                          label=ref31h, orientation = 1);
mesh3 Th31b = movemesh23(Thsq1, transfo=[XX1, YY1, ZZ1min],
                          label=ref31b, orientation = -1);
```

example40.edp code

```
verbosity=2;
load "msh3"
load "tetgen"
load "medit"

// front (z=0) and back (z=1.5) faces
real x0, x1, y0, y1;
x0=1.;
x1=2.;
y0=0.;
y1=2*pi;
mesh Thsq1 = square(5, 35, [x0 + (x1-x0)*x, y0 + (y1-y0)*y] );

func ZZ1min = 0;
func ZZ1max = 1.5;
func XX1 = x;
func YY1 = y;

int[int] ref31h = [0, 12];
int[int] ref31b = [0, 11];

mesh3 Th31h = movemesh23(Thsq1, transfo=[XX1, YY1, ZZ1max],
                          label=ref31h, orientation = 1);
mesh3 Th31b = movemesh23(Thsq1, transfo=[XX1, YY1, ZZ1min],
                          label=ref31b, orientation = -1);
```

example40.edp code, cont'd

```
// bottom (y=0) and top (y=2*pi) faces
x0 = 1.;
x1 = 2.;
y0 = 0.;
y1 = 1.5;
mesh Thsq2 = square(5, 8, [x0 + (x1-x0)*x, y0 + (y1-y0)*y] );

func ZZ2 = y;
func XX2 = x;
func YY2min = 0.;
func YY2max = 2*pi;

int[int] ref32h = [0, 13];
int[int] ref32b = [0, 14];

mesh3 Th32h = movemesh23(Thsq2, transfo=[XX2, YY2max, ZZ2],
                label=ref32h, orientation= -1);
mesh3 Th32b = movemesh23(Thsq2,transfo=[XX2,YY2min,ZZ2],
                label=ref32b, orientation= 1);
```

example40.edp code, cont'd

```
func XX3min = 1.;
func XX3max = 2.;
func YY3 = x;
func ZZ3 = y;

int[int] ref33h = [0,15];
int[int] ref33b = [0,16];

mesh3 Th33h = movemesh23(Thsq3, transfo = [XX3max, YY3, ZZ3],
                label=ref33h, orientation=1);
mesh3 Th33b = movemesh23(Thsq3, transfo = [XX3min, YY3, ZZ3],
                label=ref33b, orientation=-1);
```

example40.edp code, cont'd

```
func XX3min = 1.;
func XX3max = 2.;
func YY3 = x;
func ZZ3 = y;

int[int] ref33h = [0,15];
int[int] ref33b = [0,16];

mesh3 Th33h = movemesh23(Thsq3, transfo = [XX3max, YY3, ZZ3],
                label=ref33h, orientation=1);
mesh3 Th33b = movemesh23(Thsq3, transfo = [XX3min, YY3, ZZ3],
                label=ref33b, orientation=-1);

// glue surfaces together to make surface of rectangular parallelopiped
mesh3 Th33 = Th31h + Th31b + Th32h + Th32b + Th33h + Th33b;
```

example40.edp code, cont'd

```
func XX3min = 1.;
func XX3max = 2.;
func YY3 = x;
func ZZ3 = y;

int[int] ref33h = [0,15];
int[int] ref33b = [0,16];

mesh3 Th33h = movemesh23(Thsq3, transfo = [XX3max, YY3, ZZ3],
                label=ref33h, orientation=1);
mesh3 Th33b = movemesh23(Thsq3, transfo = [XX3min, YY3, ZZ3],
                label=ref33b, orientation=-1);

// glue surfaces together to make surface of rectangular parallelopiped
mesh3 Th33 = Th31h + Th31b + Th32h + Th32b + Th33h + Th33b;

medit("glumesh",Th33);  // plot using medit

plot(Th33);             // plot using FreeFem++
```

example40.edp code, cont'd

```
// build a mesh of a axis parallel box with TetGen
//                    x    y   z    attr  max vol
real[int] domaine = [1.5, pi, 0.75, 145, 0.0025];
```

example40.edp code, cont'd

```
// build a mesh of a axis parallel box with TetGen
//                    x    y    z    attr  max vol
real[int] domaine = [1.5, pi, 0.75, 145, 0.0025];

// Tetrahedralize the interior of the cube with tetgen
mesh3 Thfinal = tetg(Th33, switch= "pqaAAYYQ", nbofregions = 1,
                  regionlist = domaine);
```

example40.edp code, cont'd

```
// build a mesh of a axis parallel box with TetGen
//                    x    y   z    attr  max vol
real[int] domaine = [1.5, pi, 0.75, 145, 0.0025];

// Tetrahedralize the interior of the cube with tetgen
mesh3 Thfinal = tetg(Th33, switch= "pqaAAYYQ", nbofregions = 1,
                     regionlist = domaine);

medit("tetg", Thfinal);
```

example40.edp code, cont'd

```
// build a mesh of a axis parallel box with TetGen
//                    x    y    z    attr  max vol
real[int] domaine = [1.5, pi, 0.75, 145, 0.0025];

// Tetrahedralize the interior of the cube with tetgen
mesh3 Thfinal = tetg(Th33, switch= "pqaAAYYQ", nbofregions = 1,
                  regionlist = domaine);

medit("tetg", Thfinal);

// build a mesh of a cylindrical shell of interior radius 1.
// and exterior radius 2 and height 1.5
func mv2x = x*cos(y);
func mv2y = x*sin(y);
func mv2z = z;
```

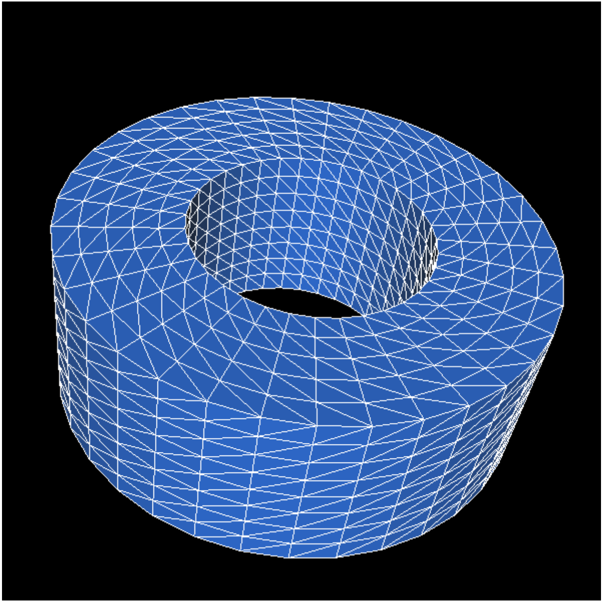
example40.edp code, cont'd

```
// build a mesh of a axis parallel box with TetGen
//                   x    y   z    attr  max vol
real[int] domaine = [1.5, pi, 0.75, 145, 0.0025];

// Tetrahedralize the interior of the cube with tetgen
mesh3 Thfinal = tetg(Th33, switch= "pqaAAYYQ", nbofregions = 1,
                  regionlist = domaine);

medit("tetg", Thfinal);

// build a mesh of a cylindrical shell of interior radius 1.
// and exterior radius 2 and height 1.5
func mv2x = x*cos(y);
func mv2y = x*sin(y);
func mv2z = z;

mesh3 Thmv2 = movemesh3(Thfinal, transfo=[mv2x, mv2y, mv2z]);
```

`example40.edp` code, cont'd

```
// build a mesh of a axis parallel box with TetGen
//                    x    y   z    attr  max vol
real[int] domaine = [1.5, pi, 0.75, 145, 0.0025];

// Tetrahedralize the interior of the cube with tetgen
mesh3 Thfinal = tetg(Th33, switch= "pqaAAYYQ", nbofregions = 1,
                  regionlist = domaine);

medit("tetg", Thfinal);

// build a mesh of a cylindrical shell of interior radius 1.
// and exterior radius 2 and height 1.5
func mv2x = x*cos(y);
func mv2y = x*sin(y);
func mv2z = z;

mesh3 Thmv2 = movemesh3(Thfinal, transfo=[mv2x, mv2y, mv2z]);

medit("cylindricalshell",Thmv2);
plot(Thmv2);
```

Example 40 result



TetGen parameters

- **`label`** = vector of integer pairs attaching new label numbers to old label numbers.
- **`switch`** = command line parameters (see below)
- **`holelist`** = Real vector of x, y, z values
- **`regionlist`** = Real vector of $x, y, z, attr, maxvol$
- **`facetcl`** = Array of (facet markers, maximum area) (used for quality of mesh)

TetGen switch parameters

- **p** = tetrahedralization of boundary
- **q** sets the “radius/edge” ratio constraint
- **a** sets volume constraints
- **A** look for attributes references in **regionlist**
- **AA** gives different labels to each region
- **r** reconstructs and refines previous mesh
- **Y** preserve mesh on exterior boundary
- **YY** preserve mesh on interior and exterior boundary
- **C** check consistence of mesh
- **CC** further consistency checks
- **V** verbose, **VV** and **VVV** more verbose
- **Q** quiet
- **M** do not merge coplanar facets
- **T** tolerance for coplanar tests
- **d** detect intersections of facets

Topics

3D

- TetGen
- Layer meshes

Mesh adaptation

Layer mesh: another way to build a mesh

- Start with a 2-D mesh
- Move it in z-direction in the interval $[z_{\min}, z_{\max}]$.
- Use **`movemesh3`** to transform it
- Can also transform it as part of **`buildlayers`**

example41.edp code

```
load "msh3"
load "medit"

verbosity = 3;
int nx=10, ny=10, nz=10; // number of steps in each direction

real x0=0., x1=1.;
real y0=0., y1=1.;
real z0=0., z1=1.;
```

example41.edp code

```
load "msh3"
load "medit"

verbosity = 3;
int nx=10, ny=10, nz=10; // number of steps in each direction

real x0=0., x1=1.;
real y0=0., y1=1.;
real z0=0., z1=1.;

// build one square
// area number = 0
mesh Thx = square(nx, ny, [x0+(x1-x0)*x, y0+(y1-y0)*y] );
```

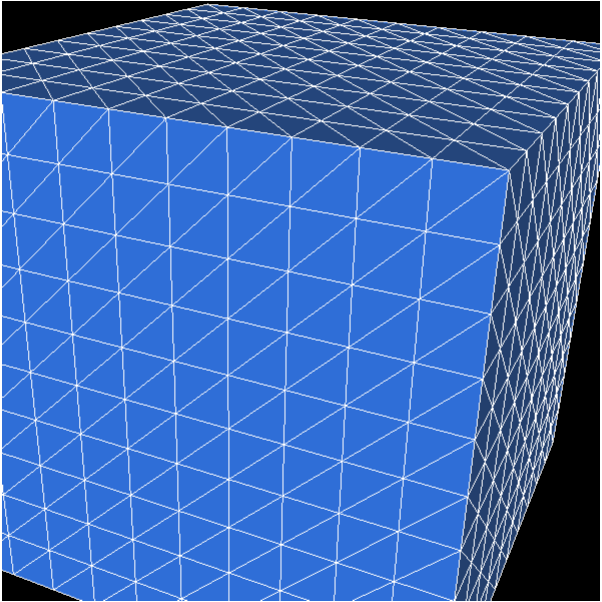
example41.edp code

```
load "msh3"
load "medit"

verbosity = 3;
int nx=10, ny=10, nz=10; // number of steps in each direction

real x0=0., x1=1.;
real y0=0., y1=1.;
real z0=0., z1=1.;

// build one square
// area number = 0
mesh Thx = square(nx, ny, [x0+(x1-x0)*x, y0+(y1-y0)*y] );

// renumber regions
// 2D: bottom = 1, right = 2, top = 3, left = 4
// Following gives pairs for top/bottom: [2D number, 3D number]
// 4 pairs for side faces
int[int] rup=[0, 6],  rdown=[0, 5],
    rside=[1, 3,  2, 2,  3, 4,  4, 1];

mesh3 Th=buildlayers(Thx, nz, zbound=[z0,z1],
       labelmid = rside, labelup = rup, labeldown = rdown);
```

example41.edp code

```
load "msh3"
load "medit"

verbosity = 3;
int nx=10, ny=10, nz=10; // number of steps in each direction

real x0=0., x1=1.;
real y0=0., y1=1.;
real z0=0., z1=1.;

// build one square
// area number = 0
mesh Thx = square(nx, ny, [x0+(x1-x0)*x, y0+(y1-y0)*y] );

// renumber regions
// 2D: bottom = 1, right = 2, top = 3, left = 4
// Following gives pairs for top/bottom: [2D number, 3D number]
// 4 pairs for side faces
int[int] rup=[0, 6],  rdown=[0, 5],
    rside=[1, 3,  2, 2,  3, 4,  4, 1];

mesh3 Th=buildlayers(Thx, nz, zbound=[z0,z1],
       labelmid = rside, labelup = rup, labeldown = rdown);

medit("Cube",Th);

plot(Th, wait=true);
```

Example 41 cube mesh



buildlayers arguments

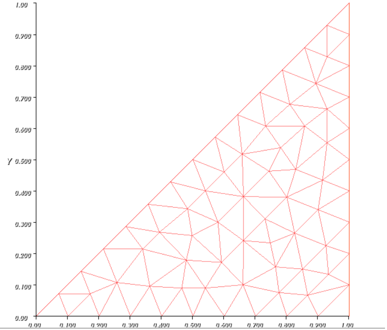
- A 2D mesh and number of layers, **M**
- **zbound** = **[zmin, zmax]** *functions* defining the lower and upper surfaces
- **coef** = *function* to introduce degenerate elements. Number of vertices associated with vertex V_i is **M*coef**(V_i). The larger the value of **coef**, the more effective levels there are over a given vertex.
- **region** = list of pairs of numbers 2D region no., 3D region no.
- **labelup**, **labeldown** are pairs of 2D label no., 3D label no.
- **labelmid**, labels of four vertical sides: pairs of 2D label no., 3D label no.

example42.edp code

```
load "msh3"
load "medit"

real RR=1, HH=1;
border Taxe(t=0, HH)x=t; y=0; label=0;
border Hypo(t=1, 0)x=HH*t; y=RR*t; label=1;
border Vert(t=0, RR)x=HH; y=t; label=2;

int nn = 10;
real h= 1./nn;
mesh Th2=buildmesh( Taxe(HH*nn) + Hypo(nn*sqrt(HH*HH + RR*RR)) + Vert(RR*nn) );
plot(Th2, wait=1);
```

example42.edp code

```
load "msh3"
load "medit"

real RR=1, HH=1;
border Taxe(t=0, HH)x=t; y=0; label=0;
border Hypo(t=1, 0)x=HH*t; y=RR*t; label=1;
border Vert(t=0, RR)x=HH; y=t; label=2;

int nn = 10;
real h= 1./nn;
mesh Th2=buildmesh( Taxe(HH*nn) + Hypo(nn*sqrt(HH*HH + RR*RR)) + Vert(RR*nn) );
plot(Th2, wait=1);
```



example42.edp code

```
load "msh3"
load "medit"

real RR=1, HH=1;
border Taxe(t=0, HH)x=t; y=0; label=0;
border Hypo(t=1, 0)x=HH*t; y=RR*t; label=1;
border Vert(t=0, RR)x=HH; y=t; label=2;

int nn = 10;
real h= 1./nn;
mesh Th2=buildmesh( Taxe(HH*nn) + Hypo(nn*sqrt(HH*HH + RR*RR)) + Vert(RR*nn) );
plot(Th2, wait=1);

int MaxLayersT=4*(int(2*pi*RR/h)/4);
func zminT = 0;
func zmaxT = 2*pi;
func fx = y*cos(z);
func fy = y*sin(z);
func fz = x;
```

example42.edp code

```
load "msh3"
load "medit"

real RR=1, HH=1;
border Taxe(t=0, HH)x=t; y=0; label=0;
border Hypo(t=1, 0)x=HH*t; y=RR*t; label=1;
border Vert(t=0, RR)x=HH; y=t; label=2;

int nn = 10;
real h= 1./nn;
mesh Th2=buildmesh( Taxe(HH*nn) + Hypo(nn*sqrt(HH*HH + RR*RR)) + Vert(RR*nn) );
plot(Th2, wait=1);

int MaxLayersT=4*(int(2*pi*RR/h)/4);
func zminT = 0;
func zmaxT = 2*pi;
func fx = y*cos(z);
func fy = y*sin(z);
func fz = x;
```

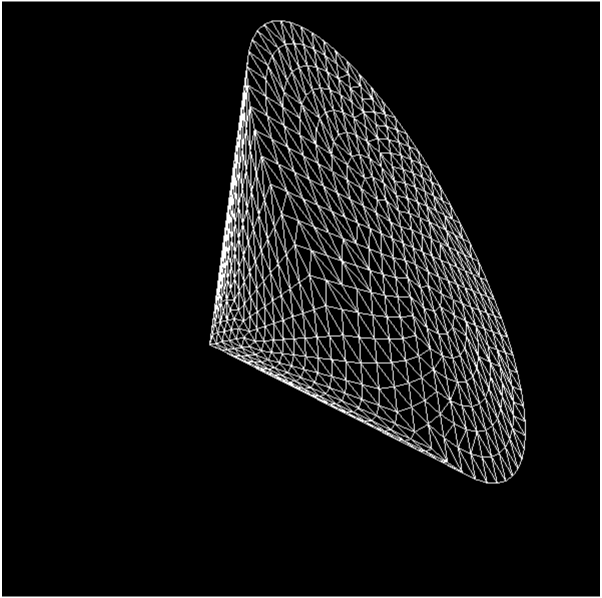
example42.edp code

```
load "msh3"
load "medit"

real RR=1, HH=1;
border Taxe(t=0, HH)x=t; y=0; label=0;
border Hypo(t=1, 0)x=HH*t; y=RR*t; label=1;
border Vert(t=0, RR)x=HH; y=t; label=2;

int nn = 10;
real h= 1./nn;
mesh Th2=buildmesh( Taxe(HH*nn) + Hypo(nn*sqrt(HH*HH + RR*RR)) + Vert(RR*nn) );
plot(Th2, wait=1);

int MaxLayersT=4*(int(2*pi*RR/h)/4);
func zminT = 0;
func zmaxT = 2*pi;
func fx = y*cos(z);
func fy = y*sin(z);
func fz = x;
int[int] r1T=[0,0], r2T=[0,0,2,2];
int[int] r4T=[0,2];

mesh3 Th3T = buildlayers(Th2, MaxLayersT, coef= max(.01,y/max(x,0.4) ),
    zbound=[zminT,zmaxT], transfo=[fx,fy,fz], facemerge=true,
    region=r1T, labelmid=r2T);
```

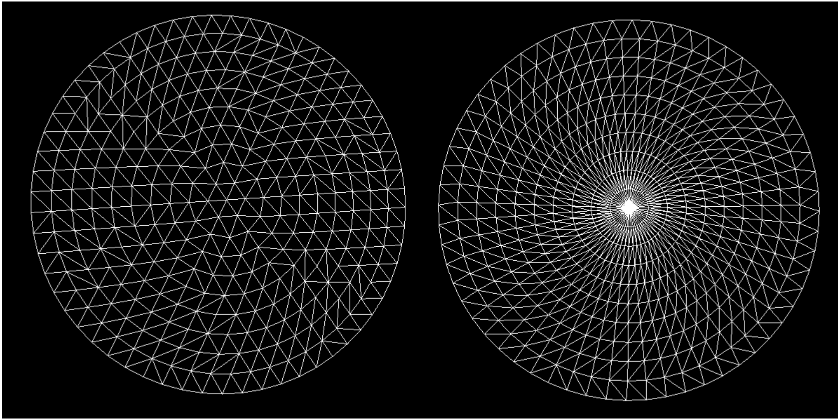
example42.edp code

```
load "msh3"
load "medit"

real RR=1, HH=1;
border Taxe(t=0, HH)x=t; y=0; label=0;
border Hypo(t=1, 0)x=HH*t; y=RR*t; label=1;
border Vert(t=0, RR)x=HH; y=t; label=2;

int nn = 10;
real h= 1./nn;
mesh Th2=buildmesh( Taxe(HH*nn) + Hypo(nn*sqrt(HH*HH + RR*RR)) + Vert(RR*nn) );
plot(Th2, wait=1);

int MaxLayersT=4*(int(2*pi*RR/h)/4);
func zminT = 0;
func zmaxT = 2*pi;
func fx = y*cos(z);
func fy = y*sin(z);
func fz = x;
int[int] r1T=[0,0], r2T=[0,0,2,2];
int[int] r4T=[0,2];

mesh3 Th3T = buildlayers(Th2, MaxLayersT, coef= max(.01,y/max(x,0.4) ),
    zbound=[zminT,zmaxT], transfo=[fx,fy,fz], facemerge=true,
    region=r1T, labelmid=r2T);

medit("cone",Th3T,wait=1);

plot(Th3T,cmm="cone");
```

Example42, mesh on cone



Example42, Effect of `coef`



`coef` value from code on left, `coef`=1 on right

Topics

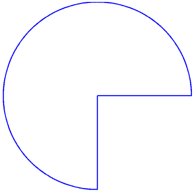
3D

- TetGen
- Layer meshes

Mesh adaptation

A test problem

- Consider a 3/4-circular region: unit disk but for $0 \leq \theta \leq 3\pi/4$



- In cylindrical coordinates

$$\Delta u = \frac{1}{r}\frac{\partial}{\partial r} r \frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2}$$

- $u = r^{2/3} \sin 2\theta/3$ satisfies Laplace's equation with Dirichlet conditions along the two straight sides.
- Singularity at origin!

Local error indicator

A local error indicator is given by

$$\eta_T = \left(h_T^2 ||f + \Delta u_h||^2_{L^2(T)} + \sum_{e \in \mathcal{E}_T} h_e \| [\frac{\partial u_h}{\partial n_k}] \|^2_{L^2(e)} \right)^{\frac{1}{2}}$$

- h_T is the longest edge of T
- $\mathcal{E}_T$ is the set of T edges not on $\Gamma = \partial\Omega$
- n_T is the outward unit normal to K
- h_e is the length of edge e
- $[g]$ is the jump of g across an edge

Example 43, adapting the mesh

1. Generate boundary and mesh
2. Define exact solution, problem, η
3. Loop
   3.1 Solve for u
   3.2 Compute true error
   3.3 Compute and plot η
   3.4 Adapt the mesh on u

example43.edp code

```
border ba(t=0,1.0   ){x=t;      y=0;      label=1;}
border bb(t=0,1.0   ){x=0;      y=-1+t;   label=2;}
border bc(t=0,3*pi/2){x=cos(t); y=sin(t); label=3;}
int n=2;
mesh Th = buildmesh (ba(4*n) + bb(4*n) + bc(30*n));

fespace Vh(Th,P1);
fespace Nh(Th,P0);
Vh u,v;
Nh eta;
real error=0.01;

func f=0;
```

example43.edp code

```
border ba(t=0,1.0   ){x=t;      y=0;      label=1;}
border bb(t=0,1.0   ){x=0;      y=-1+t;   label=2;}
border bc(t=0,3*pi/2){x=cos(t); y=sin(t); label=3;}
int n=2;
mesh Th = buildmesh (ba(4*n) + bb(4*n) + bc(30*n));

fespace Vh(Th,P1);
fespace Nh(Th,P0);
Vh u,v;
Nh eta;
real error=0.01;

func f=0;
func exactf=(y<-1.e-10? (x^2+y^2)^(1./3.)*sin( 2.*(2*pi+atan2(y,x))/3. ):
                        (x^2+y^2)^(1./3.)*sin( 2.*(     atan2(y,x))/3. ));
```

example43.edp code

```
border ba(t=0,1.0   ){x=t;       y=0;      label=1;}
border bb(t=0,1.0   ){x=0;       y=-1+t;   label=2;}
border bc(t=0,3*pi/2){x=cos(t); y=sin(t); label=3;}
int n=2;
mesh Th = buildmesh (ba(4*n) + bb(4*n) + bc(30*n));

fespace Vh(Th,P1);
fespace Nh(Th,P0);
Vh u,v;
Nh eta;
real error=0.01;

func f=0;
func exactf=(y<-1.e-10? (x^2+y^2)^(1./3.)*sin( 2.*(2*pi+atan2(y,x))/3. ):
                        (x^2+y^2)^(1./3.)*sin( 2.*(     atan2(y,x))/3. ));

Vh truerror, exactu=exactf;
plot(exactu,wait=true);
```

example43.edp code

```
border ba(t=0,1.0   ){x=t;      y=0;      label=1;}
border bb(t=0,1.0   ){x=0;      y=-1+t;   label=2;}
border bc(t=0,3*pi/2){x=cos(t); y=sin(t); label=3;}
int n=2;
mesh Th = buildmesh (ba(4*n) + bb(4*n) + bc(30*n));

fespace Vh(Th,P1);
fespace Nh(Th,P0);
Vh u,v;
Nh eta;
real error=0.01;

func f=0;
func exactf=(y<-1.e-10? (x^2+y^2)^(1./3.)*sin( 2.*(2*pi+atan2(y,x))/3. ):
                        (x^2+y^2)^(1./3.)*sin( 2.*(     atan2(y,x))/3. ));

Vh truerror, exactu=exactf;
plot(exactu,wait=true);

problem Problem1(u,v,solver=UMFPACK) =
    int2d(Th, qforder=5)( dx(u)*dx(v) + dy(u)*dy(v)) + on(1,2,3, u=exactf);

varf indicator2(unused,chiK) =
     intalledges(Th)(chiK*lenEdge*square( jump( N.x*dx(u) + N.y*dy(u) )))
    +int2d(Th)( chiK*square( hTriangle*(f + dxx(u) + dyy(u))) );
```

example43.edp code

```
border ba(t=0,1.0   ){x=t;       y=0;      label=1;}
border bb(t=0,1.0   ){x=0;       y=-1+t;   label=2;}
border bc(t=0,3*pi/2){x=cos(t); y=sin(t); label=3;}
int n=2;
mesh Th = buildmesh (ba(4*n) + bb(4*n) + bc(30*n));

fespace Vh(Th,P1);
fespace Nh(Th,P0);
Vh u,v;
Nh eta;
real error=0.01;

func f=0;
func exactf=(y<-1.e-10? (x^2+y^2)^(1./3.)*sin( 2.*(2*pi+atan2(y,x))/3. ):
                        (x^2+y^2)^(1./3.)*sin( 2.*(     atan2(y,x))/3. ));

Vh truerror, exactu=exactf;
plot(exactu,wait=true);

problem Problem1(u,v,solver=UMFPACK) =
    int2d(Th, qforder=5)( dx(u)*dx(v) + dy(u)*dy(v)) + on(1,2,3, u=exactf);

varf indicator2(unused,chiK) =
     intalledges(Th)(chiK*lenEdge*square( jump( N.x*dx(u) + N.y*dy(u) )))
    +int2d(Th)( chiK*square( hTriangle*(f + dxx(u) + dyy(u))) );
```

example43.edp code, cont'd

```
for (int i=0;i< 5;i++){
   Problem1;
   cout « u[].min « " " « u[].max « endl;
   plot(u, cmm="solution", wait=true);
   truerror = u - exactu;
   real normerror = int2d(Th)( truerror^2 );
   real normsoln = int2d(Th)( u^2 );
   plot(truerror, cmm="true error", wait=true, value=true, fill=true, nbiso=20);
   cout « " true rel error=" « normerror/normsoln « endl;
```

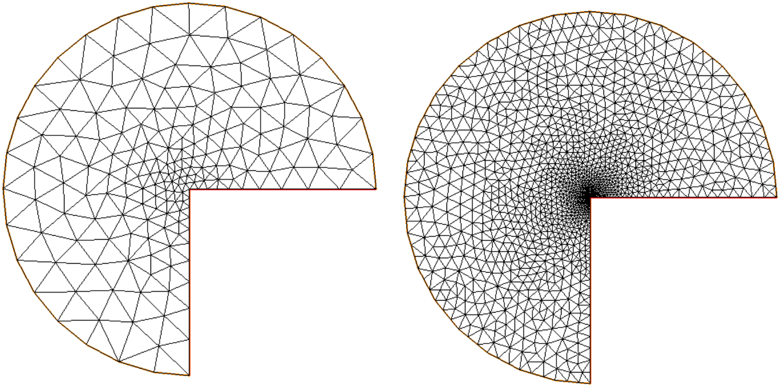
example43.edp code, cont'd

```
for (int i=0;i< 5;i++){
   Problem1;
   cout « u[].min « " " « u[].max « endl;
   plot(u, cmm="solution", wait=true);
   truerror = u - exactu;
   real normerror = int2d(Th)( truerror^2 );
   real normsoln = int2d(Th)( u^2 );
   plot(truerror, cmm="true error", wait=true, value=true, fill=true, nbiso=20);
   cout « " true rel error=" « normerror/normsoln « endl;

   cout « " indicator2 " « endl;
   eta[] = indicator2(0,Nh);
   eta=sqrt(eta);
   cout « "eta =   min " « eta[].min « " max=" « eta[].max « endl;
   plot(eta, fill=true, wait=true, cmm="indicator density ",
          value=true, nbiso=20);
```

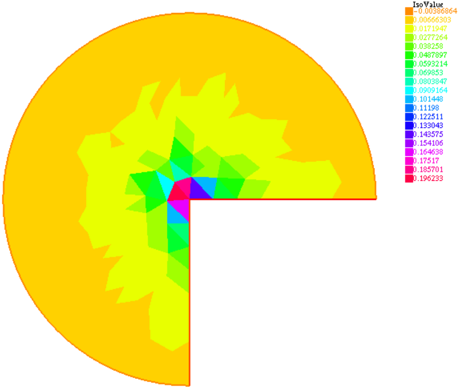
example43.edp code, cont'd

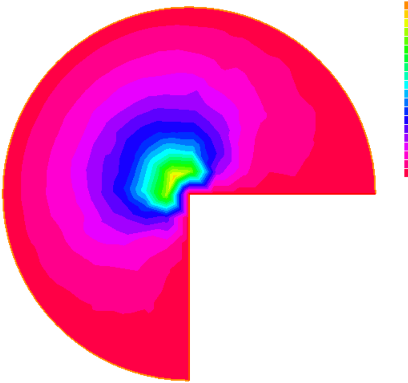
```
for (int i=0;i< 5;i++){
   Problem1;
   cout « u[].min « " " « u[].max « endl;
   plot(u, cmm="solution", wait=true);
   truerror = u - exactu;
   real normerror = int2d(Th)( truerror^2 );
   real normsoln = int2d(Th)( u^2 );
   plot(truerror, cmm="true error", wait=true, value=true, fill=true, nbiso=20);
   cout « " true rel error=" « normerror/normsoln « endl;

   cout « " indicator2 " « endl;
   eta[] = indicator2(0,Nh);
   eta=sqrt(eta);
   cout « "eta =   min " « eta[].min « " max=" « eta[].max « endl;
   plot(eta, fill=true, wait=true, cmm="indicator density ",
          value=true, nbiso=20);

   Th=adaptmesh(Th, u, err=error, anisomax=1);
   plot(Th,wait=1);
   u = u;
   eta = eta;
   error = error/2;
}
```

Example 43: Meshes

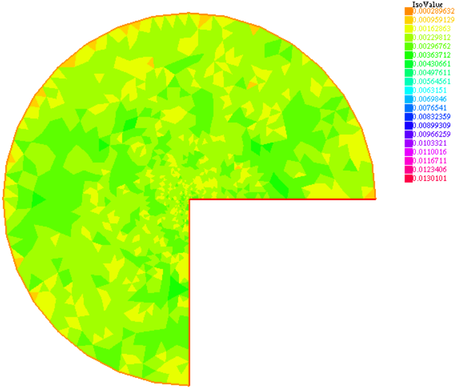


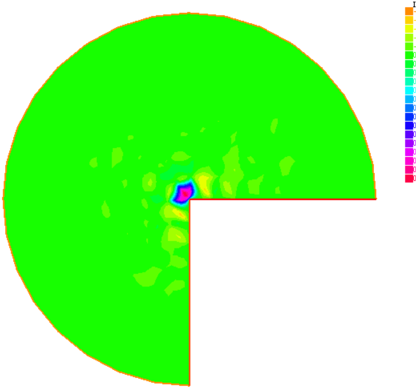
Example 43: First indicator and true error





Example 43: final indicator and true error





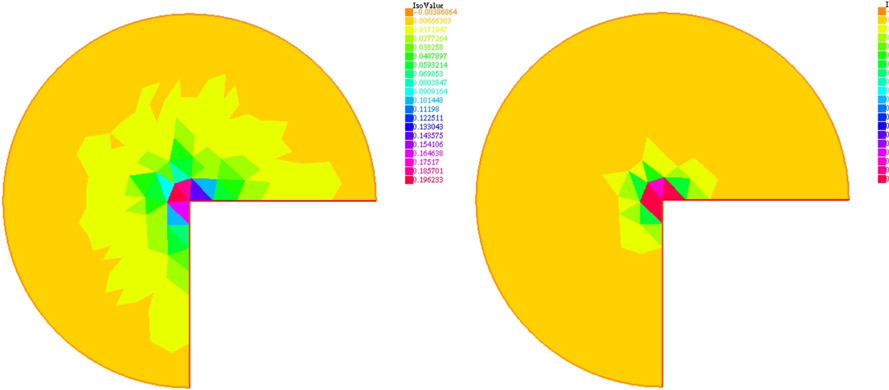
Example 43: selected printed output

```
  -  mesh:  Nb of Triangles =    588, Nb of Vertices 333
 true rel error=2.96617e-05
 indicator2
eta =   min 0.00139719 max=0.190967

  -  mesh:  Nb of Triangles =    322, Nb of Vertices 184
 true rel error=6.57629e-06
 indicator2
eta =   min 0.00709164 max=0.119733

  -  mesh:  Nb of Triangles =    627, Nb of Vertices 345
 true rel error=7.97066e-06
 indicator2
eta =   min 0.00074159 max=0.0544662

  -  mesh:  Nb of Triangles =   1386, Nb of Vertices 742
 true rel error=8.81045e-06
 indicator2
eta =   min 0.00139323 max=0.0264413

  -  mesh:  Nb of Triangles =   2671, Nb of Vertices 1408
 true rel error=9.36967e-06
 indicator2
eta =   min 0.000624381 max=0.0126753
```

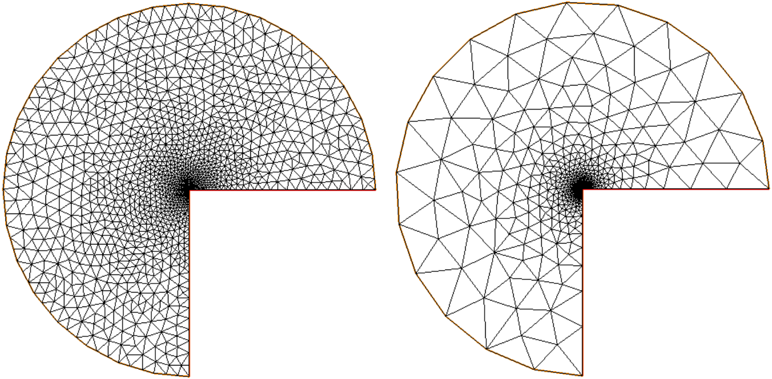
Example 44: P2 and refine on ∇u

- Use **`P2`** elements for solution
- Refine using **`[dx(u),dy(u)]`** rather than **`u`**

First indicator: 43 and 44



Final meshes: 43 and 44



Example 44: selected printed output

```
  -  mesh:  Nb of Triangles =    588, Nb of Vertices 333
 true rel error=1.83115e-06
 indicator2
eta =   min 3.32476e-05 max=0.181251
  -  mesh:  Nb of Triangles =    605, Nb of Vertices 329
 true rel error=7.29378e-07
 indicator2
eta =   min 8.63491e-05 max=0.0506088
  -  mesh:  Nb of Triangles =   1681, Nb of Vertices 882
  - Solve :
 true rel error=7.35067e-07
 indicator2
eta =   min 7.2703e-06 max=0.0116626
  number of required edges : 0
  -  mesh:  Nb of Triangles =   3643, Nb of Vertices 1880
  - Solve :
 true rel error=7.36122e-07
 indicator2
eta =   min 4.9985e-07 max=0.00320423
  -  mesh:  Nb of Triangles =   6669, Nb of Vertices 3416
```

Example 45: Using the η indicator for refinement

- Same Laplace equation as before
- Heavy use of macros
  - **`MeshSizecomputation`** to get an average *h* for current mesh
  - **`ReMeshIndicator`** to remesh using η indicator

MeshSizecomputation

```
// macro the get the current mesh size
// parameter
//  in:  Th the mesh
//       Vh  P1 fespace on Th
//  out :
// h: the Vh finite element finite set to the current mesh size
macro  MeshSizecomputation(Th,Vh,h){
  /* Th mesh
  Vh  P1 finite element space
  h   the P1 mesh size value */
```

MeshSizecomputation

```
// macro the get the current mesh size
// parameter
//  in:  Th the mesh
//       Vh  P1 fespace on Th
//  out :
// h: the Vh finite element finite set to the current mesh size
macro  MeshSizecomputation(Th,Vh,h){
  /* Th mesh
  Vh  P1 finite element space
  h   the P1 mesh size value */

  real[int]  count(Th.nv);
  // mesh size  (lenEdge = integral of 1 over all edges)
  varf vmeshsizen(u,v)=intalledges(Th,qfnbpE=1)(v);
  // number of edge / par vertex
  varf vedgecount(u,v)=intalledges(Th,qfnbpE=1)(v/lenEdge);
```

MeshSizecomputation

```
// macro the get the current mesh size
// parameter
//  in:  Th the mesh
//       Vh  P1 fespace on Th
//  out :
// h: the Vh finite element finite set to the current mesh size
macro  MeshSizecomputation(Th,Vh,h){
  /* Th mesh
  Vh  P1 finite element space
  h   the P1 mesh size value */

  real[int]  count(Th.nv);
  // mesh size  (lenEdge = integral of 1 over all edges)
  varf vmeshsizen(u,v)=intalledges(Th,qfnbpE=1)(v);
  // number of edge / par vertex
  varf vedgecount(u,v)=intalledges(Th,qfnbpE=1)(v/lenEdge);
  /*
        computation of the mesh size
        ------------------- */
  count=vedgecount(0,Vh);
  h[]=0.;
  h[]=vmeshsizen(0,Vh);
  cout << " count min = "<< count.min « " " << count.max << endl;
  h[]=h[]./count;
  cout << " - bound meshsize = " <<h[].min « " " << h[].max << endl;
} // end of macro MeshSizecomputation
```

ReMeshIndicator

- macro to remesh according the residual indicator
- In: Th the mesh
- In: Ph P0 fespace on Th
- In: Vh P1 fespace on Th
- In: vindicator the varf of to evaluate the indicator squared
- In: coef on eta

ReMeshIndicator

```
macro  ReMeshIndicator(Th,Ph,Vh,vindicator,coef) {
  Vh h=0;
  /*evalutate the mesh size  */
  MeshSizecomputation(Th,Vh,h);
```

ReMeshIndicator

```
macro  ReMeshIndicator(Th,Ph,Vh,vindicator,coef) {
  Vh h=0;
  /*evalutate the mesh size  */
  MeshSizecomputation(Th,Vh,h);
  Ph etak;
  etak[]=vindicator(0,Ph);
  cout << " global  Eta  : " << sqrt(etak[].sum) <<
            "  ......... " <<  Th.nv« endl;
  etak[]=sqrt(etak[]);
  plot(etak,cmm="arei-etak",fill=1,value=1);
  real etastar= coef*(etak[].sum/etak[].n);
  cout << " etastar = " << etastar << " sum=" << etak[].sum
       << " " << endl;
```

ReMeshIndicator

```
macro  ReMeshIndicator(Th,Ph,Vh,vindicator,coef) {
  Vh h=0;
  /*evalutate the mesh size  */
  MeshSizecomputation(Th,Vh,h);
  Ph etak;
  etak[]=vindicator(0,Ph);
  cout << " global  Eta  : " << sqrt(etak[].sum) <<
          "  .........  " <<  Th.nv« endl;
  etak[]=sqrt(etak[]);
  plot(etak,cmm="arei-etak",fill=1,value=1);
  real etastar= coef*(etak[].sum/etak[].n);
  cout << " etastar = " << etastar << " sum=" << etak[].sum
       << " " << endl;

  /* here etaK is discontinous
  we use the P1 L2 projection with mass lumping . */
  Vh fn,sigma;
  varf veta(unused,v)=int2d(Th)(etak*v);
  varf vun(unused,v)=int2d(Th)(1*v);
  fn[]  = veta(0,Vh);
  sigma[]= vun(0,Vh);
  fn[]= fn[]./ sigma[];
  fn =  max(min(fn/etastar,3.),0.3333) ;
```

ReMeshIndicator

```
macro  ReMeshIndicator(Th,Ph,Vh,vindicator,coef) {
  Vh h=0;
  /*evalutate the mesh size  */
  MeshSizecomputation(Th,Vh,h);
  Ph etak;
  etak[]=vindicator(0,Ph);
  cout << " global  Eta  : " << sqrt(etak[].sum) <<
           "  ......... " <<  Th.nv« endl;
  etak[]=sqrt(etak[]);
  plot(etak,cmm="arei-etak",fill=1,value=1);
  real etastar= coef*(etak[].sum/etak[].n);
  cout << " etastar = " << etastar << " sum=" << etak[].sum
       << " " << endl;

  /* here etaK is discontinous
  we use the P1 L2 projection with mass lumping . */
  Vh fn,sigma;
  varf veta(unused,v)=int2d(Th)(etak*v);
  varf vun(unused,v)=int2d(Th)(1*v);
  fn[]  = veta(0,Vh);
  sigma[]= vun(0,Vh);
  fn[]= fn[]./ sigma[];
  fn =  max(min(fn/etastar,3.),0.3333) ;

  /* new mesh size */
  h = h / fn ;
  Th=adaptmesh(Th,IsMetric=1,h,splitpbedge=1,nbvx=10000);
} // EOM
```

example45.edp code

```
// mesh definition
border ba(t=0,1.0)x=t;   y=0;  label=1;
border bb(t=0,1.0)x=0;   y=-1+t;  label=2;
border bc(t=0,3*pi/2)x=cos(t); y=sin(t); label=3;
int n=2;
mesh Th = buildmesh (ba(4*n) + bb(4*n) + bc(30*n));
```

example45.edp code

```
// mesh definition
border ba(t=0,1.0)x=t;   y=0;   label=1;
border bb(t=0,1.0)x=0;   y=-1+t;  label=2;
border bc(t=0,3*pi/2)x=cos(t); y=sin(t); label=3;
int n=2;
mesh Th = buildmesh (ba(4*n) + bb(4*n) + bc(30*n));

// FE space definition --
fespace Vh(Th,P1); // for the mesh size
fespace Ph(Th,P0); // for the indicator
```

example45.edp code

```
// mesh definition
border ba(t=0,1.0)x=t;   y=0;  label=1;
border bb(t=0,1.0)x=0;   y=-1+t;  label=2;
border bc(t=0,3*pi/2)x=cos(t); y=sin(t); label=3;
int n=2;
mesh Th = buildmesh (ba(4*n) + bb(4*n) + bc(30*n));

// FE space definition --
fespace Vh(Th,P1); // for the mesh size
fespace Ph(Th,P0); // for the indicator

real hinit=0.2; //
Vh   h=hinit; // the FE fonction  for the mesh size
// to build a mesh with a given mesh size  : meshsize
Th=adaptmesh(Th,h,IsMetric=1,splitpbedge=1,nbvx=10000);
plot(Th,wait=1);
```

example45.edp code

```
// mesh definition
border ba(t=0,1.0)x=t;   y=0;   label=1;
border bb(t=0,1.0)x=0;   y=-1+t;  label=2;
border bc(t=0,3*pi/2)x=cos(t); y=sin(t); label=3;
int n=2;
mesh Th = buildmesh (ba(4*n) + bb(4*n) + bc(30*n));

// FE space definition --
fespace Vh(Th,P1); // for the mesh size
fespace Ph(Th,P0); // for the indicator

real hinit=0.2; //
Vh   h=hinit; // the FE fonction  for the mesh size
// to build a mesh with a given mesh size  : meshsize
Th=adaptmesh(Th,h,IsMetric=1,splitpbedge=1,nbvx=10000);
plot(Th,wait=1);

Vh u,v;

func f=0;
func exactf=(y<-1.e-10? (x^2+y^2)^(1./3.)*sin( 2.*(2*pi+atan2(y,x))/3. ):
                        (x^2+y^2)^(1./3.)*sin( 2.*(     atan2(y,x))/3. ));

Vh truerror, exactu=exactf;
```

example45.edp code

```
problem Poisson(u,v,solver=UMFPACK) =
    int2d(Th,qforder=5)( dx(u)*dx(v) + dy(u)*dy(v))
  + on(1,2,u=exactf) + on(3,u=exactf);

varf indicator2(unused,chiK) =
     intalledges(Th)(chiK*lenEdge*square(jump(N.x*dx(u)+N.y*dy(u))))
    +int2d(Th)(chiK*square(hTriangle*(f+dxx(u)+dyy(u))) );
```

example45.edp code

```
problem Poisson(u,v,solver=UMFPACK) =
    int2d(Th,qforder=5)( dx(u)*dx(v) + dy(u)*dy(v))
  + on(1,2,u=exactf) + on(3,u=exactf);

varf indicator2(unused,chiK) =
     intalledges(Th)(chiK*lenEdge*square(jump(N.x*dx(u)+N.y*dy(u))))
    +int2d(Th)(chiK*square(hTriangle*(f+dxx(u)+dyy(u))) );

for (int i=0;i< 10;i++){
  u=u;
  Poisson;



  truerror = u - exactu;
  real normerror = int2d(Th)( truerror^2 );
  real normsoln = int2d(Th)( u^2 );
```

example45.edp code

```
problem Poisson(u,v,solver=UMFPACK) =
    int2d(Th,qforder=5)( dx(u)*dx(v) + dy(u)*dy(v))
  + on(1,2,u=exactf) + on(3,u=exactf);

varf indicator2(unused,chiK) =
     intalledges(Th)(chiK*lenEdge*square(jump(N.x*dx(u)+N.y*dy(u))))
    +int2d(Th)(chiK*square(hTriangle*(f+dxx(u)+dyy(u))) );

for (int i=0;i< 10;i++){
  u=u;
  Poisson;



  truerror = u - exactu;
  real normerror = int2d(Th)( truerror^2 );
  real normsoln = int2d(Th)( u^2 );



  real cc=0.7;
  if(i>5) cc=1;
  if(i<9) {
    ReMeshIndicator(Th,Ph,Vh,indicator2,cc);
  }
```

example45.edp code

```
problem Poisson(u,v,solver=UMFPACK) =
    int2d(Th,qforder=5)( dx(u)*dx(v) + dy(u)*dy(v))
  + on(1,2,u=exactf) + on(3,u=exactf);

varf indicator2(unused,chiK) =
     intalledges(Th)(chiK*lenEdge*square(jump(N.x*dx(u)+N.y*dy(u))))
    +int2d(Th)(chiK*square(hTriangle*(f+dxx(u)+dyy(u))) );

for (int i=0;i< 10;i++){
  u=u;
  Poisson;
  plot(Th,u,wait=1);
  cout « u[].min « " " « u[].max « endl;
  plot(u,cmm="solution",wait=1);
  truerror = u - exactu;
  real normerror = int2d(Th)( truerror^2 );
  real normsoln = int2d(Th)( u^2 );
  plot(truerror,cmm="true error",wait=true,value=true,fill=true);
  cout « " true rel error=" « normerror/normsoln « endl;

  real cc=0.7;
  if(i>5) cc=1;
  if(i<9) {
    ReMeshIndicator(Th,Ph,Vh,indicator2,cc);
  }
  plot(u,Th,cmm="remeshed solution",wait=1,value=1);
}
```

Example 45: a refined mesh