An introduction to scientific computing using free software FreeFem++

Ionut Danaila[1] and Frédéric Hecht[2]

[1]Université de Rouen, France

[2]Université Pierre et Marie Curie, Paris

Lesson 1: March 7, 2016.

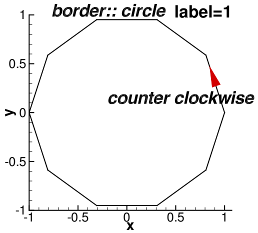
Outline of this Lesson

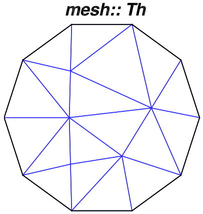
Building a mesh with FreeFem++ (v01)

Any computation starts with a mesh

mesh_circle_v1.edp

```
/* Mesh of a circle */

// Parameters

int nbseg=10;
real R=1, xc=0, yc=0;

// border
border circle(t=0,2*pi){label=1;
                        x=xc+R*cos(t);
                        y=yc+R*sin(t);}
plot(circle(nbseg),cmm="border");

// FE mesh
mesh Th = buildmesh(circle(nbseg));
plot(Th, cmm="mesh of a circle");
```



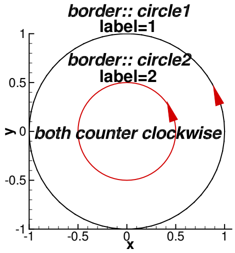


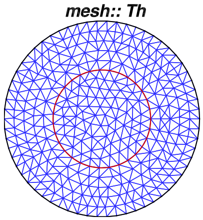
Building a mesh with FreeFem++ (v02)

A mesh with a sub-domain:: + circle2(nbseg*2*pi*R2)

mesh_circle_v2.edp

```
/* Mesh of a circle with a subdomain */
// Parameters
int nbseg=10;real R=1, xc=0, yc=0, R2=R/2;
// borders
border circle1(t=0,2*pi){label=1;
                        x=xc+R*cos(t);
                        y=yc+R*sin(t);}
border circle2(t=0,2*pi){label=2;
                        x=xc+R2*cos(t);
                        y=yc+R2*sin(t);}
plot(circle1(nbseg*2*pi*R)+circle2(nbseg*2*pi*R2)
    ,cmm="border");
// FE mesh
mesh Th = buildmesh(circle1(nbseg*2*pi*R)
                   +circle2(nbseg*2*pi*R2));
plot(Th, cmm="mesh of a circle with subdomain");
// Identify subdomains
cout <<"inner region:: number ="<<
  Th(xc,yc).region <<endl;
cout <<"inner region:: number ="<<
  Th(xc+(R2+R)/2,yc).region <<endl;
```



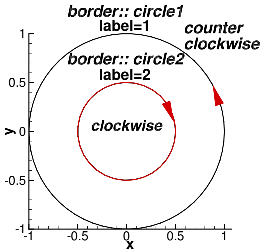


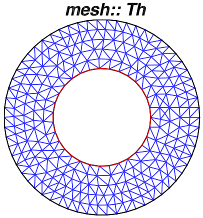
Building a mesh with FreeFem++ (v03)

A mesh with a hole inside:: + circle2(-nbseg*2*pi*R2)

mesh_circle_v3.edp

```
/* Mesh of a circle with a hole inside */
// Parameters
int nbseg=10;
real R=1, xc=0, yc=0, R2=R/2;
// border
border circle1(t=0,2*pi){label=1;
                         x=xc+R*cos(t);
                         y=yc+R*sin(t);}
border circle2(t=0,2*pi){label=2;
                         x=xc+R2*cos(t);
                         y=yc+R2*sin(t);}
plot(circle1(nbseg*2*pi*R)+circle2(nbseg*2*pi*R2)
    ,cmm="border");
// FE mesh
mesh Th = buildmesh(circle1(nbseg*2*pi*R)
                    +circle2(-nbseg*2*pi*R2));
plot(Th, cmm="mesh of a circle with subdomain");
```





Building a mesh with FreeFem++ (v04)

A mesh with a hole inside:: using macros to avoid bugs

be carreful with the syntax of EndOfMacro and inside comments

mesh_circle_v4.edp

```
/* Mesh of a circle with a hole inside
   using macros    */

macro Bcircle(bname,Rm,xm,ym,labelm)
      /* circle border */
      border bname(t=0,2*pi)
      {label=labelm; x=xm+Rm*cos(t);y=ym+Rm*sin(t);}//EOM
// Parameters
int nbseg=10;
real R=1, xc=0, yc=0, R2=R/2;

// borders
Bcircle(circle1,R ,xc,yc,1);
Bcircle(circle2,R2,xc,yc,2);

plot(circle1(nbseg*2*pi*R)+circle2(nbseg*2*pi*R2),cmm="border");
// FE mesh
mesh Th = buildmesh(circle1(nbseg*2*pi*R)
                    +circle2(-nbseg*2*pi*R2));
plot(Th, cmm="mesh of a circle with subdomain");
```

Building a mesh with FreeFem++ (v05)

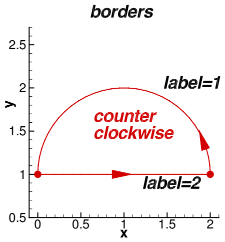
Mesh for a half-disk::

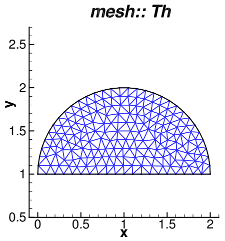
check intersection points

oriented borders (counter clockwise)

mesh_circle_v5.edp

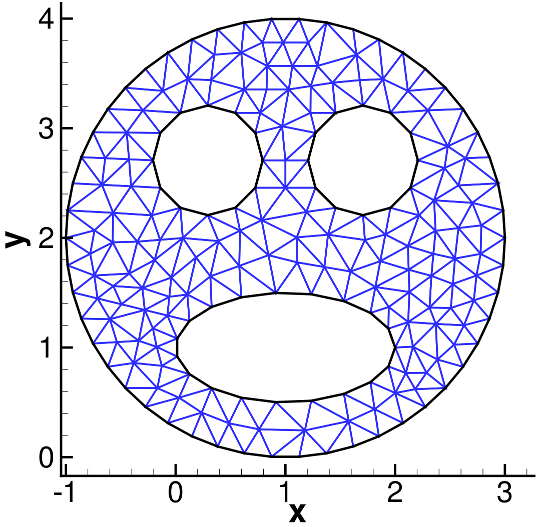
```
/* Mesh of a half-disk  */
// Parameters
int nbseg=10;
real R=1, xc=1, yc=1;
// border
border Dcircle(t=0, pi)     {label=1;
                            x=xc+R*cos(t);
                            y=yc+R*sin(t);}
border Daxis  (t=xc-R,xc+R){label=2;
                            x=t;
                            y=yc;}
plot(Dcircle(nbseg*pi*R)+Daxis(nbseg*2*R),cmm="
    border");
// FE mesh
mesh Th = buildmesh(Dcircle(nbseg*pi*R)+Daxis(
    nbseg*2*R));
plot(Th, cmm="mesh of a half-disk");
```





Building a mesh with FreeFem++ (v06)

Intermission: Mesh of a smiley



Building a smiley with FreeFem++ (v06)

mesh_circle_v6.edp

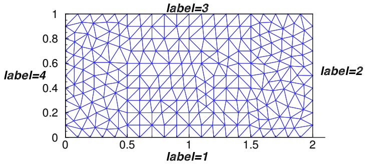
```
/* Mesh of a smiley using macros     */
macro Bellipse(bname,Rmx,Rmy,xm,ym,labelm)
      border bname(t=0,2*pi)
      {label=labelm; x=xm+Rmx*cos(t);y=ym+Rmy*sin(t);}//EOM
// Parameters
int nbseg=10;
//head
real Rh=2, xh=1, yh=2, Lh=2*pi*Rh;
Bellipse(bs1,Rh,Rh,xh,yh,1);
//eyes
real xy1=xh+Rh/2*cos(pi/4), yy=yh+Rh/2*sin(pi/4), Ry=Rh/4, Ly=2*pi*Ry;
Bellipse(bs2,Ry,Ry,xy1,yy,2);
real xy2=xh-Rh/2*cos(pi/4);
Bellipse(bs3,Ry,Ry,xy2,yy,3);
//mouth
real a=Rh/2, b=Rh/4, Lm=pi*sqrt(2*(a^2+b^2));
Bellipse(bs4,a,b,xh+0,yh-Rh/2,4);

plot(bs1(nbseg*Lh)+bs2(nbseg*Ly)+bs3(nbseg*Ly)+bs4(nbseg*Lm));
// FE mesh
mesh Th = buildmesh(bs1(nbseg*Lh)+bs2(-nbseg*Ly)+bs3(-nbseg*Ly)+bs4(-
    nbseg*Lm));
plot(Th, cmm="mesh of a smiley");
```

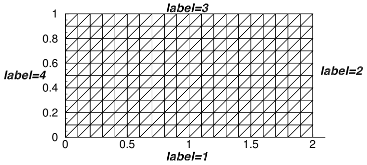
Building the mesh of a rectangle (using ”square”)

Mesh a rectangle::
using the built-in function ”square”

mesh_rectangle_v02.edp

```
/* Mesh of a rectangle using square
    function  */
// Parameters
int nbseg=10;
real L=2,H=1;
real xc1=0,    yc1=0;
// FE mesh
mesh Th = square(nbseg*L,
                 nbseg*H,
        [xc1+x*L,yc1+y*H]);
plot(Th, cmm="mesh of a rectangle");
```





Building the mesh of a rectangle (using macros)

mesh_rectangle_v01.edp

```
/* Mesh of a rectangle   */
macro Bsegment(bname,xP1,yP1,xP2,yP2,Ls,labelm)
      real Ls=sqrt((xP1-xP2)^2+(yP1-yP2)^2);
      border bname(t=0,Ls)
      {label=labelm; x=xP1+t*(xP2-xP1)/Ls;y=yP1+t*(yP2-yP1)/Ls;}//EOM
// Parameters
int nbseg=10;
real L=2,H=1;
real xc1=0,    yc1=0,
     xc2=xc1+L,yc2=yc1,
     xc3=xc2,  yc3=yc2+H,
     xc4=xc1,  yc4=yc3;
//borders
Bsegment(bs1,xc1,yc1,xc2,yc2,Ls1,1);
Bsegment(bs2,xc2,yc2,xc3,yc3,Ls2,2);
Bsegment(bs3,xc3,yc3,xc4,yc4,Ls3,3);
Bsegment(bs4,xc4,yc4,xc1,yc1,Ls4,4);
plot(bs1(nbseg*Ls1)+bs2(nbseg*Ls2)+bs3(nbseg*Ls3)+bs4(nbseg*Ls4));
// FE mesh
mesh Th = buildmesh(bs1(nbseg*Ls1)+bs2(nbseg*Ls2)+bs3(nbseg*Ls3)+bs4(
    nbseg*Ls4));
plot(Th, cmm="mesh of a rectangle");
```