BAMG: Bidimensional Anisotropic Mesh Generator

Frédéric Hecht *

draft version v1.00 decembre 2006

The software **bamg** is a Bidimensional Anisotropic Mesh Generator, It a part of FreeFem++ software `www.freefem.org/ff++`

1 Outline of the commands

This software is activated by the command `bamg` or by the `bamg-g` command for graphical and debugging version.

This software can

- create a mesh from a geometry.
  - input file with `-g filename` (file type **DB mesh**).
  - output file arguments `-oxxxx filename` where `xxxx` is the type of output file (see point 1.4 for more details).
  - smoothing, quad, utility and other parameters

  In addition, metric specification may be prescribed with the help of `-M filename` argument (file type **Metric**).

  All the options are described below.

- adapt a mesh from a background mesh using a metric file, or solution file
  - input files `-b filename` where the suffixe of the file define the type of the file `.amdba`, `.am_fmt`, `.am`, `.ftq`, `.nopo` and otherwise the file is a **BD mesh** file.
  - `-M filename` (file type **Metric**)or `-MBB filename` or `-Mbb filename` (solution file type **BB** or **bb**)
  - metric control parameters, `-err val`, `-errg val` , ...
  - output file arguments reqired
  - interpolation, smoothing, quad, utility parameter. and other parameter

- to do smoothing, to make quadrilaterals, to split internal edge with 2 boundary vertices, ... an existing mesh (in this case the metric is use to change the definition of the element's quality).

*University Pierre et Marie Curie, LJLL, BC187 4 Place Jussieu 75252 PARIS cedex 05 FRANCE, email: Frederic.Hecht@upmc.fr, http://www-rocq.inria.fr/Frederic.Hecht

-- input files `-r filename` (file type **DB mesh**).

-- `-M filename` (file type **Metric**).

-- output file `-o filename` (file type **DB mesh**).

-- some other parameter:

  – `-thetaquad val` To create quad with 2 triangles
  – `-2` To create the submesh with mesh size $h = h/2$
  – `-2q` To split all triangles in 3 quad. and to split all quad. in 4 quad.
  – `-NbSmooth ival` To change the number of smoothing iteration (3 by default if the metric field is set with arguments : `-M` or `-Mbb`, 0 otherwise.
  – ...

- to just construct just a metric file, if you have an other mesher

-- input files `-r filename` (file type **DB mesh**).

-- `-Mbb filename` or `-MBB filename`

-- + all the arguments of the metric construction

-- output file `-oM filename` (file type **Metric**)

- to just do the P^1 interpolation of the solution

-- input files `-r filename` (file type **DB mesh**).

-- `-rBB filename` or `-rbb filename`

-- output file `-wBB filename` or `-wbb filename`

Others options are described below.

2 Detailed list of arguments

2.1 Input Mesh (the three arguments are exclusive)

`-g filename` Set the geometry for mesh generation (file type **DB mesh**).

`-b filename` Set the background mesh for mesh adaption, the suffixe of the file name set the type of the inputs file `.amdba, .am, .am_fmt, .ftq, .msh, .nopo` give the type of the file otherwise the type is (file type **DB mesh**).

`-r filename` Set the mesh for mesh transformation (require `-M` or `-Mbb` arguments). Remark: the geometry is the background geometry. If the background geometry is not define by a **BD mesh**, then we reconstruct a geometry from the boundary mesh and the thetamax angle.

`-thetamax (float)` to define the AngleOfCornerBound in degree which say if you have a corner or not (cf. 4.1), if the geometry is reconstruct, ie. the input mesh have no geometrical definition.

2.2 Metric Definition (or mesh size definition)

`-M filename` Set the metric which is defined on the background mesh or on the geometry. (see description of geometry data structure). (file type **Metric**)

A metric field can be constructed from a solution defined on the background mesh. Solution (or solution components) are supposed to be of continuous P^1 Lagrange type.

This option is activated by

`-Mbb filename` (where the file is of type **bb**, see description of data structure **bb**).

`-MBB filename` (where the file is of type **BB**, see description of data structure **BB**).

When more of one argument `-M` and `-Mbb` , `-MBB` are present, the resulting metric is the intersection of the given and constructed metrics.

In addition, some parameters may act on the metric.

`-coef val` Set the value of a multiplicative coefficient on the mesh size.

`val` is of type double precision and default value is 1.0

`-ratio val` Set the ratio for a prescribed smoothing on the metric. If `val` is 0 (default value) or less than 1.1 no smoothing on the metric is done.

If `val` > 1.1 the speed of mesh size variation is bounded by $log($`val`$)$. Remark: As `val` is closer to `1`, the number of vertices generated increases. This may be useful to control the thickness of refined regions near shocks or boundary layers.

`-hmin val` Set the value of the minimal edge size. (`val` is of type double precision and default value is related to the size of the domain to be meshed and the precision of mesh generator).

`-hmax val` Set the value of the maximal edge size. (`val` is of type double precision and default value is the diameter of the domain to be meshed)

`-errg val` Set the value of the relative error on geometry, by default this error is 0.1, and in any case this value is geater than $1/\sqrt{2}$. Remark that mesh size created by this option can be smaller than the `-hmin` argument due to geometrical constraint.

When we construct the metric (`-Mbb` argument or `-MBB` argument) , additional parameters can be used.

The metric is constructed from the Hessian $\mathcal{H}$ of the solution η read. If we read more than one solution, the intersection of all metric associated to each solution is realized to get the final metric.

The parameters `err, coef, CutOff` present in formula (1) and (2) are prescribed via `-err, -coef, -CutOff` arguments respectively.

`-err val` Set the level of the P^1 interpolation error. (The present `val` parameter is a double precision parameter and the default value for `err` is 10^{-2}.)

- `-iso` Enforce the metric to be isotropic
- `-aniso` The metric may be of anisotropic type (default case).
- `-RelError` The metric is evaluated using the criterium of equirepartion of relative error. In this case the metric is defined by

$$\mathcal{M} = \left(\frac{1}{\texttt{err coef}^2} \quad \frac{|\mathcal{H}|}{max(\texttt{CutOff}, |\eta|)} \right)^p \tag{1}$$

- `-AbsError` The metric is evaluated using the criterium of equidistribution error. In this case the metric is define by

$$\mathcal{M} = \left(\frac{1}{\texttt{err coef}^2} \quad \frac{|\mathcal{H}}{sup(\eta) - inf(\eta)} \right)^p . \tag{2}$$

- `-CutOff val` Set the limit value of the relative error evaluation (parameter is double precision parameter and default value for `CutOff` is 10^{-5}).
- `-power val` Set the power parameter p in metric construction in the equations 1 and 2, the value is 1 by default.
- `-NbJacobi ival` Set the number of iterations in a smoothing procedure during the metric construction, 0 imply no smoothing. (`ival` is an integer parameter, with default value is 1).
- `-maxsubdiv val` Change the metric such that the maximal subdivision of a background's edge is bound by the `val` number (alway limited by 10).
- `-anisomax val` Set the bound of mesh anisotropy with respect to minimal h^{Min} mesh size in all direction so the maximal h^{Max} mesh size in all direction is bound by `val`$\times h^{Min}$.

2.3 Smoothing, quad, and Utility Parameter

- `-NbSmooth ival` Set the number of iterations of the smoothing procedure, (`ival` $=$ 3 by default).
- `-omega val` Set the relaxation parameter of the smoothing procedure, (`val` $=$ 1.8 by default).
- `-splitpbedge` Split in two all internal edges with two boundary vertices (cf. fig. 3).
- `-nosplitpbedge` Don't split internal edges with two boundary vertices, default case (cf. fig. 2).
- `-thetaquad val` To create quad with 2 triangles,if the 4 angles of a quad are in $[val, 180 - val]$.
- `-2` To create the submesh with mesh size $h = h/2$, use with `-coef 2` to get the right mesh size
- `-2q` To split all triangles in 3 quad. and to split all quad, use with `-coef 2` to get the right mesh size

2.4 Output Meshes (always required in case of mesh generation)

The generated mesh(es) are written in file(s) where the output mesh files is defined by

- `-o filename` Create a (file type **DB mesh**).
- `-oamdba filename` Create a amdba file.
- `-oftq filename` Create a ftq file.
- `-omsh filename` Create a msh file (FreeFem3 file).
- `-oam_fmt filename` Create a am_fmt file.
- `-oam filename` Create a am file.
- `-onopo filename` Create a nopo file (link with Modulef library).

`-o mfilename` Set the name of the output file (file type **DB mesh**).

In addition, for compatibility with existing applications which use other data structures for describing input meshes, the following options are provided.

`-oamdba filename` The data describing the generated mesh are stored in the file `filename` which is of type **amdba** (see description of such structure below).

`-oam_fmt filename` The data describing the generated mesh are stored in the file `filename` which is of type **am_fmt** (see description of such structure below).

2.5 Interpolation Options

In the adaption process, a solution has been computed with the background mesh. In order to transfer the solution (of the problem under consideration) on the new generated mesh, an interpolation of old solution is necessary. This tranferred solution may be a good initial guess for the solution on the new mesh. This interpolation is carried out in a P^1 Lagrange context.

`-rbb filename` specify the solution to be interpolated on the new mesh. (the file is of type **bb**) and the default file is the same as the solution file for the metric construction (see `-Mbb` arguments)

`-wbb filename` specify the name of the file where the interpolated solution is stored. (file is of type **bb**)

2.6 Other Parameters

`-v ival` Set the level of printing (verbosity , which can be chosen between 0 and 10 default value is 1).

`-nbv ival` or `-nbs ival` Set the maximal number of vertices generated by the mesh generator (the `ival` is an integer and the default value is 50 000)

Remark: you can put all this arguments in a file: `DATA_bamg` these option is useful on MacIntosh because we have not other way to pass easily arguments to a program.

3 Examples

3.1 Creation of Mesh from a Geometry

Only two arguments are required:

`-g gfilename` which describe the geometry is **DB Mesh** format.

`-o filename` output file which will contain the created mesh in **DB Mesh** format.

3.1.1 Example Square $]-1,1[^2$ with Mesh Size 0.666 at Vertices

The `gfilename (square_g.msh)` contents

```
MeshVersionFormatted 0
Dimension 2
Vertices 4
 -1 -1 1
 1 -1 2
 1 1 3
 -1 1 4
Edges 4
1 2 1
2 3 1
3 4 2
4 1 2
hVertices
0.666 0.666 0.666 0.666
```

The Unix command is

```
 bamg  -o square_g.msh -o square_0.msh
```

The result mesh is show in Fig. 1 and the output is:

```
--- Construction of a mesh from the geometry : square_g.msh
     with less than 50000 vertices, write on square_0.msh
 Cpu for read the geometry 0s
 -- Triangles::NewPoints Nb sommets = 17 Nb triangles = 20
                         NbSwap final = 0 Nb Total Of Swap = 13
 Cpu for meshing 0.11s
```

and the create file `square_0.msh` content:

```
MeshVersionFormatted 0

Dimension
2

Identifier
"G=square_g.msh, Date: 97/06/26 11:33 55s"
```

```
Geometry
"square_g.msh"

Vertices
17
-1 -1 1
1 -1 2
1 1 3
-1 1 4
-0.33333337307 -1 1
0.33333325386 -1 1
1 -0.33333337307 1
1 0.33333325386 1
0.33333337307 1 2
-0.33333325386 1 2
-1 0.33333337307 2
-1 -0.33333325386 2
0.111111083171 -0.333333333333 0
-0.111111085034 0.333333333333 0
-0.444444458414 -0.333333294218 0
0.555555540654 -4.00468707085e-08 0
0.444444456552 0.499999979045 0

Edges
12
1 5 1
5 6 1
6 2 1
2 7 1
7 8 1
8 3 1
3 9 2
9 10 2
10 4 2
4 11 2
11 12 2
12 1 2

Triangles
20
12 15 11 1
13 5 6 1
4 11 10 1
1 5 15 1
16 7 8 1
14 10 11 1
6 2 7 1
14 9 10 1
15 12 1 1
3 9 17 1
5 13 15 1
6 7 13 1
```

```
16 14 13 1
17 9 14 1
14 15 13 1
14 11 15 1
7 16 13 1
8 3 17 1
16 17 14 1
16 8 17 1

SubDomainFromMesh
1
3 1 1 1

VertexOnGeometricVertex
4
 1 1
 2 2
 3 3
 4 4

VertexOnGeometricEdge
8
 5 1 0.333333308498
 6 1 0.666666646798
 7 2 0.333333308498
 8 2 0.666666646798
 9 3 0.333333308498
 10 3 0.666666646798
 11 4 0.333333308498
 12 4 0.666666646798

EdgeOnGeometricEdge
12
1 1
2 1
3 1
4 2
5 2
6 2
7 3
8 3
9 3
10 4
11 4
12 4

End
```

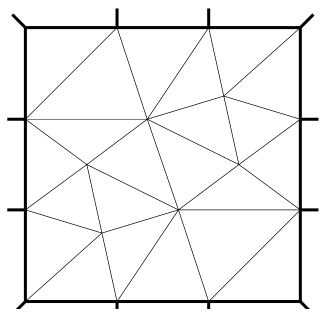


Figure 1: The mesh of square $]-1,1[^2$ with mesh size 0.666 at each vertices.

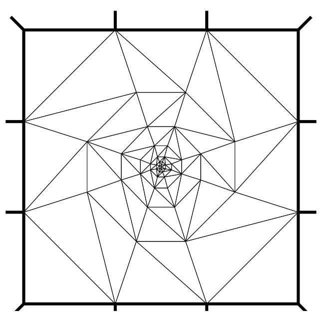


Figure 2: The mesh of square $]-1,1[^2$ with a refinement and with the default option `-nosplitpbedge` (see Page 2).

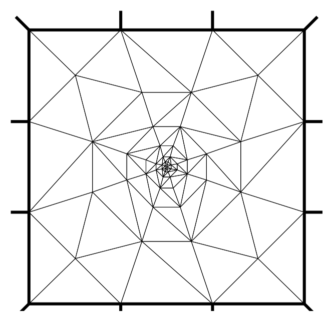


Figure 3: The mesh of square $]-1,1[^2$ with a refinement and with the option `-splitpbedge` (see Page 2).

3.1.2 Example Square $]-1,1[^2$ with a Refinement

Taken from the previous example, we want to have a refinement with mesh size of about $h = 0.001$ in the neighborhood of $(0,0)$. This vertex has to be prescribed in the geometry file.

The `gfilename (square_raf_g.msh)` contents

```
MeshVersionFormatted 0
Dimension 2
Vertices 5
 -1 -1 1
 1 -1 2
 1 1 3
 -1 1 4
0 0 0
Edges 4
1 2 1
2 3 1
3 4 2
4 1 2
hVertices
0.666 0.666 0.666 0.666 0.01
#  if you do not say that the vertex number 5 is required
#  the mesher lost this information so no refinement
#  around the vertex (0.,0.) is done

RequiredVertices 1
5
```

The two Unix command are

`bamg  -o square_raf_g.msh -o square_raf.msh` (Fig. 2)

`bamg   -splitpbedge -o square_raf_g.msh -o square_raf.msh` (Fig. 3)

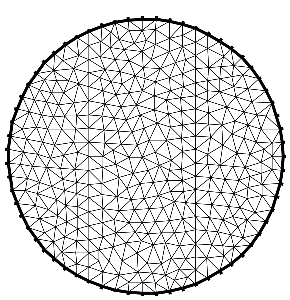


Figure 4: The mesh of the unit circle defined from an octagon, `AngleOfCornerBound` $> 45°$

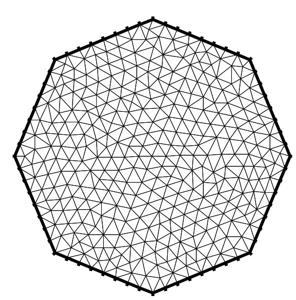


Figure 5: The mesh of an octagon, `AngleOfCornerBound` $< 45°$

3.1.3 Example of a Circle

The `gfilename (circle_g.msh)` contents

```
MeshVersionFormatted 0
AngleOfCornerBound 46
Dimension 2
Vertices 8
1 0 1
.7071067811865476 .7071067811865476 1
0 1 1
-.7071067811865476 .7071067811865476 1
-1 0 1
-.7071067811865476 -.7071067811865476 1
0 -1 1
.7071067811865476 -.7071067811865476 1

Edges 8
1 2 1
2 3 1
3 4 1
4 5 1
5 6 1
6 7 1
7 8 1
8 1 1

hVertices
0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
```

The Unix command is

```
bamg  -o circle_g.msh -o circle_0.msh
```

If you change the value of `AngleOfCornerBound` from 46 to 44, in the file `circle_g.msh`, you obtain the mesh of an octagon (see Fig. 5), because the angle of the consecutive normal on octagone is 45° and if you set the `MaximalAngleOfCorner` to a number less than 45°, then the software find eight corners, otherwise it don't find any corner (see Fig. 4) .

3.2 Mesh adaption loop to minimize the interpolation error

let be f_1 and f_2 two given fonction, the problem is to find the coarsed mesh such that the error L^∞ of the interpolation for the 2 fonctions, is bounded with ε. The two error associated to two founctions are denote by ε_1 and ε_2 .

The example `dotest1.pl` in directory `examples/test` is a `perl` script to solve this problem.

In this example the 2 fonctions are:

$$f_1 = (10 * x^3 + y^3) + atan2(0.001, (sin(5 * y) - 2 * x)),$$
$$f_2 = (10 * y^3 + x^3) + atan2(0.01, (sin(5 * x) - 2 * y)).$$

and the domain is unit circle.

The first coefficient in $atan2$ function modelise the thickness of a boundary layer. In this case we do 20 iterations and the constant parameters of bamg are:

```
-AbsError -NoRescaling -NbJacobi 2 -NbSmooth 5 -hmax 2 -hmin 0.0000005
-ratio 0 -nbv 100000 -v 4 -err 0.05 -errg 0.01
```

You can see on figures 6, 7, 8, 9 the convergence of the process. This was acheved near the iteration 11, and also the converge is more faster on ε_2 rather ε_1. This is normal because the thickness of a boundary layer is an order 10 beetwen f_1 and f_2.

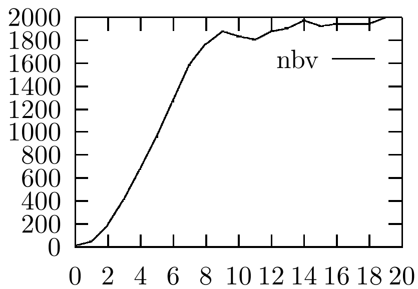


Figure 6: Nb of vertices versus iteration

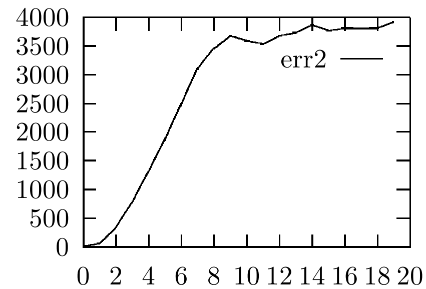


Figure 7: Nb of traingles versus iteration

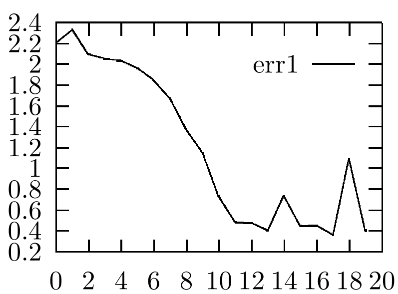


Figure 8: ε_1 versus iteration

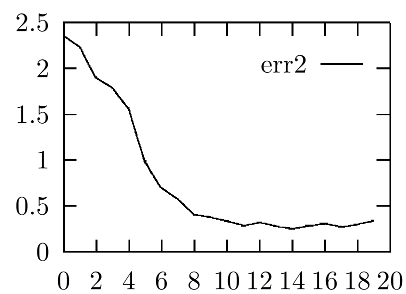
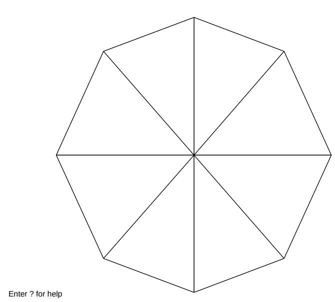
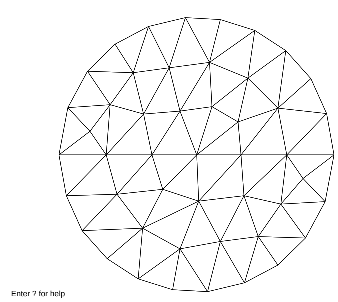


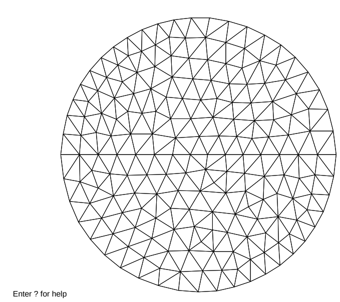
Figure 9: ε_2 versus iteration



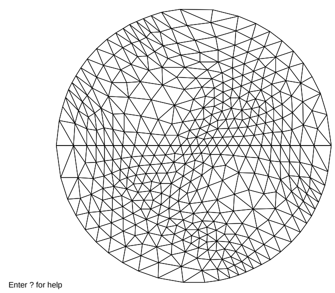
0, ε =2.20 2.35, 8T 9V



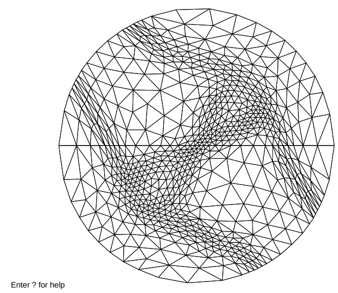
1, ε =2.33 2.22, 72T 48V



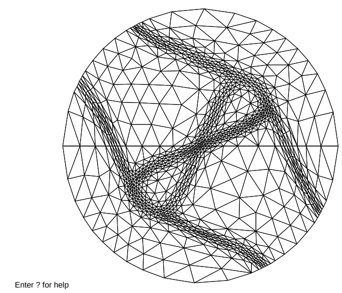
2, ε =2.09 1.89, 337T 192V



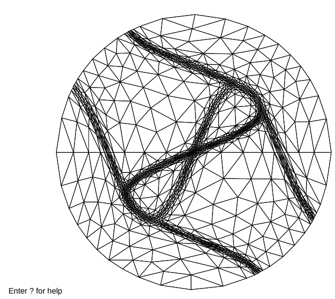
3, ε =2.05 1.78, 780T 420V



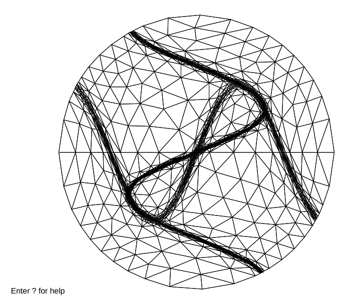
4, ε =2.03 1.55, 1300T 681V



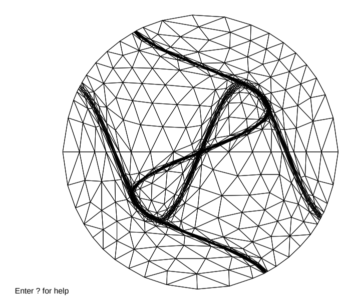
5, ε =1.95 0.97, 1867T 966V



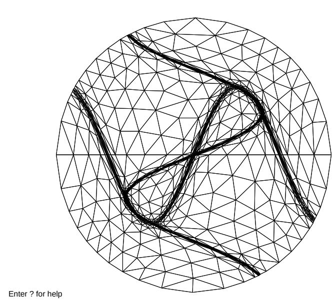
6, ε =1.84 0.69, 2475T 1271V



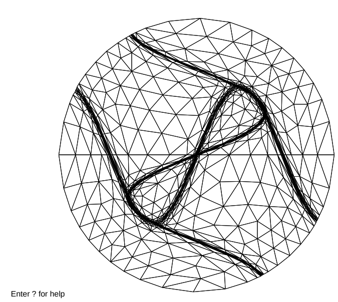
7, ε =1.66 0.57, 3106T 1587V



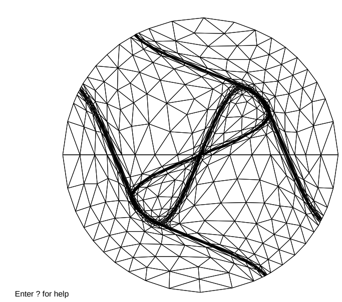
8, ε =1.37 0.40, 3459T 1764V



9, ε =1.15 0.37, 3681T 1876V

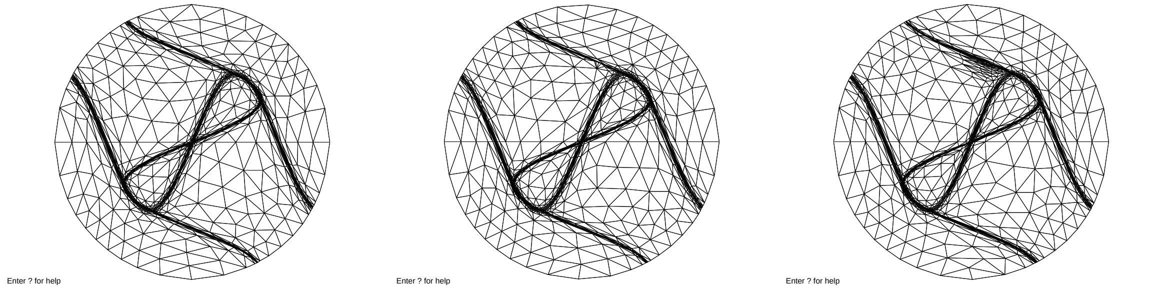


10, ε =0.73 0.33, 3588T 1831V

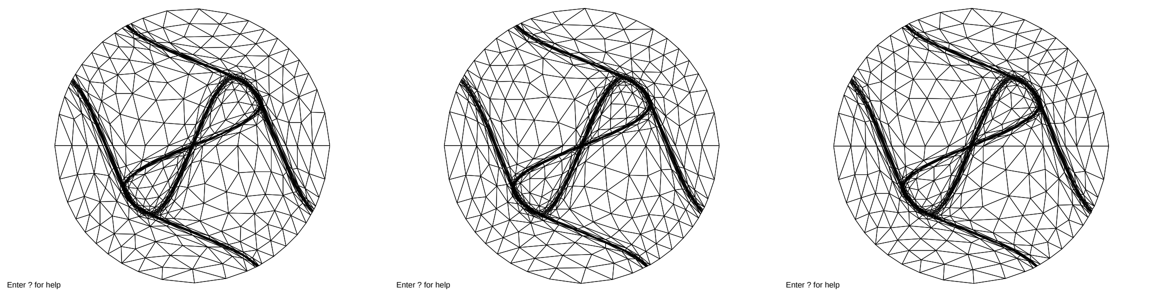


11, ε =0.48 0.28, 3530T 1803V

Figure 10: Firsr mesh to 8Th mesh



12, ε =0.47 0.32, 3675T 1875V 13, ε =0.40 0.27, 3728T 1903V 14, ε =0.73 0.24, 3866T 1971V



15, ε =0.44 0.28, 3765T 1921V 16, ε =0.45 0.30, 3814T 1945V 17, ε =0.35 0.27, 3797T 1936V

Figure 11: 9Th mesh to 17Th Mesh

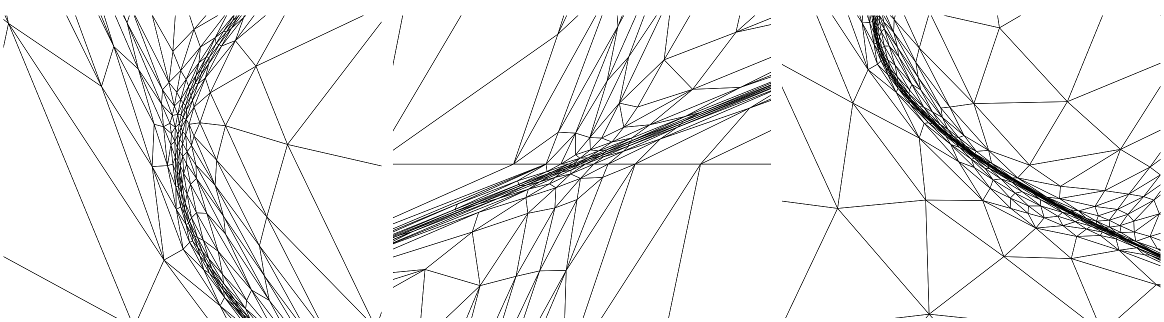


Figure 12: three zoom of the 17th mesh around intersection of boundary layer

3.3 Mesh adaption loop to compute a flow around a NACA0012 wing

In this section we make the computation of Euler flow around a wing at Mach number 0.8, with the NSC2KE software get from:

ftp://ftp.inria.fr/INRIA/Projects/Gamma/NSC2KE.tar.gz

The half wing is defined analyticaly for $t \in [0, 1.008930411365]$ by

$$x(t) = \frac{t}{1.00893041136}$$

$$y(t) = 5 * .12 * (0.2969\sqrt{t} - 0.126t - 0.3516t^2 + 0.2843t^3 - 0.1015t^4)$$

The initial mesh (geometry) is created with the following `awk` script (where `awk` is a common unix command) with 40 vertices and egdes on the wing and eight vertices on outside boundary (circle with radius 5 and with center $(1, 0)$)

```
END {
Pi=3.14159265358979;
i20=20;
i8=8;
r = 5;
c0x = 1;
c0y=0;
print "Dimension",2;
print "MaximalAngleOfCorner 46";


print "Vertices",i20+i20+i8;

# the vertices on naca012 wing (clock wise)
 for (i=-i20;i<i20;i++) {
   x = i/i20;
   x = x^4;
   t = 1.008930411365*x;
   y = 5*.12*(0.2969*sqrt(t) - 0.126*t - 0.3516*t^2 + 0.2843*t^3 - 0.1015*t^4);
   if(i<0) y=-y;
   print x,y,3;}

# vertices on circle  (counter clock wise)
 for (i=0;i<i8;i++) {
   t=i*Pi*2/i8;
   print c0x+r*(cos(t)),c0y+r*sin(t),5;}


 print "Edges",i20+i20+i8;

#  Edges on wing
 k = 1
 j = i20+i20-1; # previous points
 for (i=0;i<i20+i20;j=i++)
   { print j+k,i+k,3;} # previous, current vertex

# Edges on circle
 k = i20+i20+1;
 j = i8-1;# previous points
 for (i=0;i<i8;j=i++)
   { print k+j,k+i,5;} # previous, current vertex

#  One subdomain, region on left side of the wing
#  because clock wise sens.
 print "SubDomain",1;
 print 2,1,1,0;
}
```

The adpation loop is done with a Unix shell script. Any modification is done in the flow solver NSC2KE software, all the transformations are done in the shell script. The input files for NSC2KE are the two files DATA and MESH, plus a file INIT_NS for restarting, and the output files are SOL_NS and RESIDUAL.

The source of the shell script is

```
#!/bin/sh -eu
#   The -e option to stop  on error
#   We use awk to do flotting point  operation in the shell
#
#   installation variable
bamg=../../bamg
NSC2KE=ns/NSC2KE

# For awk because in french the number 1/1000 is written 0,001 not 0.001.
# To be sure the RADIXCHAR  is '.'   (cf.  Native Language Support, do  man locale)
LANG=C
export LANG

# Some VAR
ifin=20
j=0
INIT=0
LastIteration=0
NBITER=500
HMIN=0.05
HMINGLOBAL=0.0005
HCOEF=0.8

# Clean of the output file
rm -f [A-Z]*

# Create the geometry file
awk -f naca.awk </dev/null >MESH_g.msh

#  create the initial mesh  MESH_0.amdba
$bamg  -g MESH_g.msh -o  MESH_$j.msh -hmax 2 -oamdba MESH_$j.amdba

while [ $j -lt $ifin ] ; do
   #  i = j + 1
   i=`expr $j + 1`

   #  LastIteration = LastIteration + NBITER
   LastIteration=`expr $LastIteration + $NBITER`


   ## set the current MESH
   rm -f  MESH
   ln -s MESH_$j.amdba MESH

   ## create the DATA file for NSC2KE form file data
   ##  change  2 lines for initialisation
   rm -f DATA
   sed -e "s/^INIT/$INIT/" -e "s/^lastIteration/$LastIteration/" <data >DATA

   echo "--------- NSC2KE iteration $j    -----------"
   $(NSC2KE)

   ##  find the nb of vertices in the file MESH
   nbv=`head -1 MESH|awk '{print $1}'`

   ##  create the bb file for interpolation
   echo "2 4 $nbv  2" > SOL_$j.bb
   cat  SOL_NS  >> SOL_$j.bb

   ## create the bb file for metric construction
```

```
   ## in file SOL_NS  on each line i we have  ro ro*u ro*v  ro*energy
   ## at vertex i
   ## + a last line with 2 number last iteration and last time
   echo "2 1 $nbv  2" > MACH.bb
   awk 'NF==4 { print sqrt($2*$2+$3*$3)/$1}' SOL_NS >>  MACH.bb

   ##  put all the residual in one file
   cat RESIDUAL >>RESIDU

   ## set HMIN = MAX($HMINGLOBAL,$HMIN*$HCOEF)
   HMIN=`awk "END {c=$HMIN*$HCOEF;c=c<$HMINGLOBAL ?$HMINGLOBAL:c; print c};" </dev/null`

   $bamg  -b MESH_$j.msh -err 0.001 -errg 0.05 -AbsError \
      -hmin $HMIN -hmax 3 -Mbb MACH.bb  -o  MESH_$i.msh \
      -oamdba MESH_$i.amdba  -ratio 2 -rbb SOL_$j.bb -wbb INIT_$i.bb

   ##  creation of the INIT_NS for NSC2KE
   ##  remove fisrt line form bb file and  add the last line of  SOL_NS
   sed 1d <INIT_$i.bb >INIT_NS
   tail -1 SOL_NS >>INIT_NS

   # change i and not initialisation
   j=$i
   INIT=1
done
```

Remark: this unix shell script is quite complicated because we need floating arithmetic and we use **awk** to do this, the file modification is not so easy at this level (We don't want to make any modification in the software `NSC2KE`).

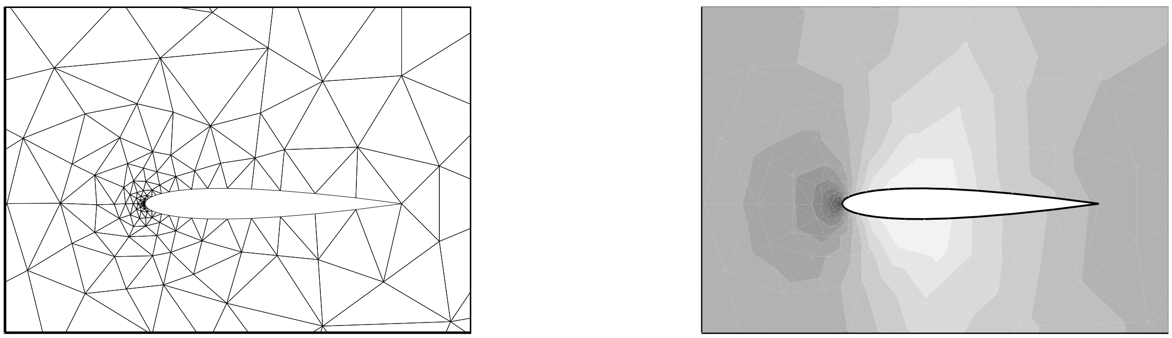
Now the real input file with two parameters `INIT` and `LastIteration` for `NSC2KE` is **data** and contains:

```
0          --> =0 2D, =1 AXISYMMETRIC
0          --> =0 Euler, =1 Navier-Stokes
1.e2       --> Reynolds by meter (the mesh is given in meter)
0.         --> inverse of Froude number (=0 no gravity)
0.8        --> inflow Mach number
1.         --> ratio pout/pin
1          --> wall =1 newmann b.c.(adiabatic wall), =2 (isothermal wall)
300.       --> inflow temperature (in Kelvin) for Sutherland laws
288.       --> if isothermal walls , wall temperature (in Kelvin)
0.0        --> angle of attack
1          --> Euler fluxes =1 roe, =2 osher,=3 kinetic
3          --> nordre = 1st order scheme, =2 2ndorder, =3 limited 2nd order
1          --> =0 global time steping (unsteady), =1 local Euler, =2 local N.S.
1.        --> cfl
LastIteration --> number of time step
500        --> frequence for the  solution to be saved
1.e10      --> maximum physical time for run (for unsteady problems)
-4.        --> order of magnitude for the residual to be reduced (for steady problems)
INIT        --> =0 start with uniform solution, =1 restart from INIT_NS
cccc  turbulence  cccccccccccccccccccccccccccccccccccccccccccccccccccc
0          --> =0 no turbulence model, =1 k-epsilon model
0          --> =0 two-layer technique, =1 wall laws
1.e-2      --> delta in wall laws or limit of the one-eq. model. (in meter)
0          --> =0 start from uniform solution for k-epsilon, =1 from INIT_KE
-1.e10 1.e10 -1.e10 1.e10    --> xtmin,xtmax,ytmin,ytmax (BOX for k-epsilon r.h.s)
```

The results of some iterations are show in Fig. 13 and 14.

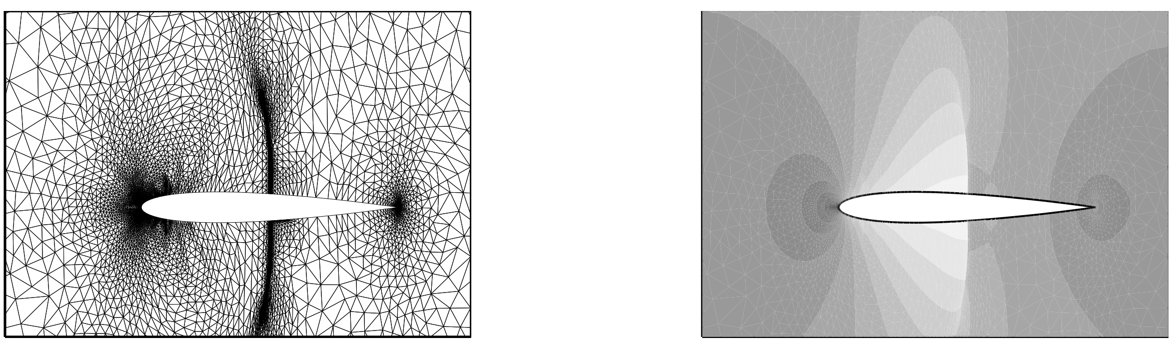
The output file of the execution on a workstation HP9000/C160 is filtered with the unix command

```
egrep 'iteration|With io|HMIN|Vertices|real|user|sys|residu|imum mach' output |\
awk -f  awk.grep-output
```



Iteration 0, 222 vertices and 388 triangles, hmin = 0.04 Iso Mach

Figure 13: initialisation (Iterations 0)



Iteration 19, 12057 vertices and 23756 triangles, hmin = 0.000576461 Iso Mach

Figure 14: Last Iteration (19)

```
     Cpu for meshing with io :    0.55s  Nb Triangles/s = 705.454545455
iter  0   nb time step  = 500    residual  =  2.85105E-07
     kt= 500  Hmim = 0.04  Mach min,max =  .70057814E-02 .11067072E+01
     Cpu time for meshing with io  : 1.34s  Nb Triangles/s = 1620.14925373
iter  1   nb time step  = 1000    residual  =  1.73430E-06
     kt= 1000  Hmim = 0.032  Mach min,max =  .30541784E+00 .12212421E+01
     Cpu time for meshing with io  : 3.02s  Nb Triangles/s = 1658.60927152
iter  2   nb time step  = 1500    residual  =  1.18474E-07
     kt= 1500  Hmim = 0.0256  Mach min,max =  .20168246E+00 .12325419E+01
     Cpu time for meshing with io  : 3.94s  Nb Triangles/s = 1474.61928934
iter  3   nb time step  = 2000    residual  =  1.52682E-07
     kt= 2000  Hmim = 0.02048  Mach min,max =  .17727955E+00 .12367692E+01
     Cpu time for meshing with io  : 4.54s  Nb Triangles/s = 1468.94273128
iter  4   nb time step  = 2500    residual  =  6.38675E-08
     kt= 2500  Hmim = 0.016384  Mach min,max =  .23734790E+00 .12410167E+01
     Cpu time for meshing with io  : 5.5s  Nb Triangles/s = 1453.45454545
iter  5   nb time step  = 3000    residual  =  1.57335E-07
     kt= 3000  Hmim = 0.0131072  Mach min,max =  .20607744E+00 .12512298E+01
     Cpu time for meshing with io  : 6.29s  Nb Triangles/s = 1404.29252782
iter  6   nb time step  = 3500    residual  =  8.24310E-08
     kt= 3500  Hmim = 0.0104858  Mach min,max =  .18735437E+00 .12603147E+01
     Cpu time for meshing with io  : 6.86s  Nb Triangles/s = 1407.43440233
iter  7   nb time step  = 4000    residual  =  1.30050E-07
     kt= 4000  Hmim = 0.00838864  Mach min,max =  .10309023E+00 .12801212E+01
     Cpu time for meshing with io  : 7.74s  Nb Triangles/s = 1290.05167959
iter  8   nb time step  = 4500    residual  =  6.85484E-08
     kt= 4500  Hmim = 0.00671091  Mach min,max =  .73364258E-01 .12616327E+01
     Cpu time for meshing with io  : 8.32s  Nb Triangles/s = 1294.23076923
iter  9   nb time step  = 5000    residual  =  7.39922E-08
     kt= 5000  Hmim = 0.00536873  Mach min,max =  .99447273E-01 .12856138E+01
     Cpu time for meshing with io  : 9.23s  Nb Triangles/s = 1274.32286024
iter  10   nb time step  = 5500    residual  =  5.10840E-08
     kt= 5500  Hmim = 0.00429498  Mach min,max =  .84549703E-01 .12803136E+01
     Cpu time for meshing with io  : 9.9s  Nb Triangles/s = 1284.54545455
iter  11   nb time step  = 6000    residual  =  2.79104E-08
     kt= 6000  Hmim = 0.00343598  Mach min,max =  .50481815E-01 .12759361E+01
     Cpu time for meshing with io  : 10.74s  Nb Triangles/s = 1256.1452514
iter  12   nb time step  = 6500    residual  =  2.30129E-08
     kt= 6500  Hmim = 0.00274878  Mach min,max =  .40456027E-01 .12978700E+01
     Cpu time for meshing with io  : 11.4s  Nb Triangles/s = 1261.92982456
iter  13   nb time step  = 7000    residual  =  1.83656E-08
     kt= 7000  Hmim = 0.00219902  Mach min,max =  .19100450E-01 .12874627E+01
     Cpu time for meshing with io  : 12.16s  Nb Triangles/s = 1237.25328947
iter  14   nb time step  = 7500    residual  =  1.44616E-08
     kt= 7500  Hmim = 0.00175922  Mach min,max =  .33476338E-01 .12832742E+01
     Cpu time for meshing with io  : 13.15s  Nb Triangles/s = 1242.96577947
iter  15   nb time step  = 8000    residual  =  1.22459E-08
     kt= 8000  Hmim = 0.00140738  Mach min,max =  .27367299E-01 .13000010E+01
     Cpu time for meshing with io  : 13.69s  Nb Triangles/s = 1266.25273923
iter  16   nb time step  = 8500    residual  =  1.06712E-08
     kt= 8500  Hmim = 0.0011259  Mach min,max =  .24571048E-01 .13177971E+01
     Cpu time for meshing with io  : 14.86s  Nb Triangles/s = 1288.15612382
iter  17   nb time step  = 9000    residual  =  7.31801E-09
     kt= 9000  Hmim = 0.00090072  Mach min,max =  .33929362E-02 .12964380E+01
     Cpu time for meshing with io  : 16.62s  Nb Triangles/s = 1254.87364621
iter  18   nb time step  = 9500    residual  =  5.45442E-09
     kt= 9500  Hmim = 0.000720576  Mach min,max =  .47151004E-02 .13105147E+01
     Cpu time for meshing with io  : 18.77s  Nb Triangles/s = 1265.63665424
iter  19   nb time step  = 10000    residual  =  4.03388E-09
     kt= 10000  Hmim = 0.000576461  Mach min,max =  .82820384E-02 .13168910E+01
     Cpu time for meshing with io  : 20.82s  Nb Triangles/s = 1285.20653218
real  2:03:58.6
user  1:43:23.8
```

We do no analysis of the results because, here we just want to show a way to make mesh adaption, not to make real computations, and that you can see after 19 iterations, the maximiun of the Mach number is not stabilized, so the computations are not really converge. In this example the user CPU time for meshing is 3mn 18.89s over a total user CPU time 1H43mn 23.8s. The ratio of CPU time for meshing over total CPU time is 3.20% in this case.

4 List of the Different Type of File

4.1 DB Mesh File

Notations. The terms in `policy` are the file items. The blanks, the <<new lines>> and the tabulations are the item separators. The comments start with the character string `#` and end at the end of the line, except if they are in a string. The comments are placed between the fields.

The notation $\big($..., i=1,n$\big)$ stands for an implicit DO loop as in FORTRAN.

The syntactic entities are field names, integer values `(I)`, (double) floating values `(R)`, strings `(C*)` (1024 characters at the best) being placed between `""`. The blanks, the <<new lines>> are significant when used between quotes and to use a quote `"` in a string, one has to type it twice `" "` (as in FORTRAN), booleans `(B)` : 0 for false, an other value for true (1 in general), numbers: for instance a vertex number is denoted by `@@Vertex`.

The entities of number type (assuming that the numbering starts from 1 as in FORTRAN) are the vertex numbers `@@Vertex`, the edge numbers `@@Edge`, the triangle numbers `@@Tria`, the quadrilateral numbers `@@Quad`, the numbers of a vertex in the appropriate support (see after), $\texttt{@@Vertex}^{supp}$, the numbers of a support edge $\texttt{@@Edge}^{supp}$, the numbers of a support triangle $\texttt{@@Tria}^{supp}$, the numbers of a support quadrangle $\texttt{@@Quad}^{supp}$,

In addition, $Ref\phi_i$ denotes a number related to physical attribute.

Description in extenso. The data structure includes at first a string identifying the release

- `MeshVersionFormatted 0`

Then, the fields are defined as follows

- `Dimension (I)` dim
- `Vertices (I)` NbOfVertices
  $\Big(\big(\texttt{(R)}\ x_i^j ,\ \text{j=1,dim} \big) ,\ \texttt{(I)}\ Ref\phi_i^v ,\ \text{i=1,NbOfVertices} \Big)$
- `Edges (I)` NbOfEdges
  $\Big(\texttt{@@Vertex}_i^1 , \texttt{@@Vertex}_i^2 ,\ \texttt{(I)}\ Ref\phi_i^e ,\ \text{i=1,NbOfEdges} \Big)$
- `Triangles (I)` NbOfTriangles
  $\Big(\big(\texttt{@@Vertex}_i^j ,\ \text{j=1,3} \big) ,\ \texttt{(I)}\ Ref\phi_i^t ,\ \text{i=1,NbOfTriangles} \Big)$
- `Quadrilaterals (I)` NbOfQuadrangles
  $\Big(\big(\texttt{@@Vertex}_i^j ,\ \text{j=1,4} \big) ,\ \texttt{(I)}\ Ref\phi_i^t ,\ \text{i=1,NbOfQuadrangles} \Big)$
- `SubDomain (I)` NbOfSubDomain

$$\Big(\ \texttt{(I)}\ type_i, \text{if} \left\{ \begin{array}{ll} type_i == 2: & \texttt{@@Edge}_i \\ type_i == 3: & \texttt{@@Tria}_i \\ type_i == 4: & \texttt{@@Quad}_i \end{array} \right\} ,\ \texttt{(I)}\ \text{Orientation}_i ,\ \texttt{(I)}\ Ref\phi_i^{sdd} , \ \text{i=1,NbOfSubDomain} \Big)$$

- `SubDomainFromGeom` `(I)` NbOfSubDomainFromGeom

$$\Big(\ \texttt{(I)}\ type_i, \text{if}\ \left\{\ type_i == 2:\ \ \texttt{@@Edge}_i\ \right\}\ ,\ \texttt{(I)}\ \text{Orientation}_i\,,\ \texttt{(I)}\ Ref\phi_i^{sdd}\,,\ \text{i=1}\,,\text{NbOfSubDomainFromGeom}\Big)$$

- `SubDomainFromMesh` `(I)` NbOfSubDomainFromMesh

$$\Big(\ \texttt{(I)}\ type_i, \text{if}\ \left\{\begin{array}{ll} type_i == 3: & \texttt{@@Tria}_i \\ type_i == 4: & \texttt{@@Quad}_i \end{array}\right\}\ ,\ \texttt{(I)}\ \text{Orientation}_i\,,\ \texttt{(I)}\ Ref\phi_i^{sdd}\,,\ \text{i=1}\,,\text{NbOfSubDomainFromMesh}\Big)$$

- `Corners` `(I)` NbOfCorners
  $\Big(\texttt{@@Vertex}_i\,,\ \text{i=1}\,,\text{NbOfCorners}\Big)$
- `RequiredVertices` `(I)` NbOfRequiredVertices
  $\Big(\texttt{@@Vertex}_i\,,\ \text{i=1}\,,\text{NbOfRequiredVertices}\Big)$
- `RequiredEdges` `(I)` NbOfRequiredEdges
  $\Big(\texttt{@@Edge}_i\,,\ \text{i=1}\,,\text{NbOfRequiredEdges}\Big)$
- `TangentAtEdges` `(I)` NbOfTangentAtEdges
  $\Big(\texttt{@@Edge}_i\,,\ \texttt{(I)}\ \text{VertexInEdge}\,,\Big(\texttt{(R)}\ \text{x}_i^j\,,\ \text{j=1,dim}\Big)\,,\ \text{i=1}\,,\text{NbOfTangentAtEdges}\Big)$
- `AngleOfCornerBound` `(R)` θ
- `Geometry`
  `(C*)` FileNameOfGeometricSupport

  - `VertexOnGeometricVertex`
    `(I)` NbOfVertexOnGeometricVertex
    $\Big(\texttt{@@Vertex}_i\,,\texttt{@@Vertex}_i^{geo}\,,\ \text{i=1}\,,\text{NbOfVertexOnGeometricVertex}\Big)$
  - `EdgeOnGeometricEdge`
    `(I)` NbOfEdgeOnGeometricEdge
    $\Big(\texttt{@@Edge}_i\,,\texttt{@@Edge}_i^{geo}\,,\ \text{i=1}\,,\text{NbOfEdgeOnGeometricEdge}\Big)$

- `MeshSupportOfVertices`
  `(C*)` FileNameOfMeshSupport

  - `VertexOnSupportVertex`
    `(I)` NbOfVertexOnSupportVertex
    $\Big(\texttt{@@Vertex}_i\,,\texttt{@@Vertex}_i^{supp}\,,\ \text{i=1}\,,\text{NbOfVertexOnSupportVertex}\Big)$
  - `VertexOnSupportEdge`
    `(I)` NbOfVertexOnSupportEdge
    $\Big(\texttt{@@Vertex}_i\,,\texttt{@@Edge}_i^{supp}\,,\ \texttt{(R)}\ u_i^{supp}\ ,\ \text{i=1}\,,\text{NbOfVertexOnSupportEdge}\Big)$

- `CrackedEdges` `(I)` NbOfCrackedEdges
  $\Big(\texttt{@@Edge}_i^1\,,\texttt{@@Edge}_i^2\,,\ \text{i=1}\,,\text{NbOfCrackedEdges}\Big)$
- `EquivalencedEdges` `(I)` NbOfEquivalencedEdges
  $\Big(\texttt{@@Edge}_i^1\,,\texttt{@@Edge}_i^2\,,\ \text{i=1}\,,\text{NbOfEquivalencedEdges}\Big)$
- `PhysicsReference` `(I)` NbOfPhysicsReference
  $\Big(\ \texttt{(I)}\ Ref\phi_i\,,\texttt{(C*)}\ \text{CommentOnThePhysic}\,,\ \text{i=1}\,,\text{NbOfPhysicsReference}\Big)$

- `IncludeFile (C*)` filename
- `BoundingBox` $\Big($ `(R)` Min_i `(R)` Max_i , i=1,dim $\Big)$

A Few Remarks. In the following, we give some comments concerning the different fields. At first, one may notice that some fields are strictly required while some others are optional[1].

The comments and remarks are given according to their introduction order in the above description.

The string `MeshVersionFormatted` indicates the release identificator and the type of the present file. `MeshVersionUnformatted` is an alternate case for this field.

The edge table, `Edges`, includes only, a priori, the edges with a significant reference number $Ref\phi$.

The elements are given with respect to their geometric nature (triangle, quadrilaterals, etc.). In this way, when several types of elements exist in the mesh, it is not required to take care of the element type and thus to have to manage a table which content (in terms of number of values) may change from one element to the other.

The sub-domains (cf. `SubDomainFromGeom` or `SubDomain`) (a connex component of the plan minus all the geometric edges) are defined using one edge in two dimensions along with an orientation information, (Orientation$_i$), indicating on which side of this entity the sub-domain lies. The sub-domain number is $Ref\phi^s$.

The mesh generator create the record `SubDomainFromMesh`, where the sub-domain is defined with a triangle or a quadrilateral and the orentation is always 1, plus the $Ref\phi^s$.

Warning: if no sub-domain are defined with `SubDomainFromGeom` or `SubDomain` then we suppose to mesh all the bounded connex component of the plan minus all the geometric edges.

Remark: the records `SubDomainFromGeom` and `SubDomain` are exclusive.

A corner point, `Corners` (for a support type structure), is a point where there is a C^0 continuity between the edges sharing the point. Thus, a corner will be necessarily a mesh vertex.

The required vertices, `RequiredVertices`, are the vertices of the support that must be present in the mesh as element vertices. Similarly, some edges or (triangular or quadrilateral) faces can be required.

The tangent vector to an edge, `TangentAtEdges`, consists to give the tangent vector (with respect to the surface) for this edge at the indicated endpoint. Giving the tangent vector of an edge by means of the tangent vector at a point enables us to deal with the case where several edges (boundary lines) are emanating from a point.

The corner threshold, `AngleOfCornerBound`, is a value enabling to decide the continuity type between two edges or two faces not clearly defined or not explicitely specified.

The mesh vertices are related to some entities of the support. There are two categories of support, a geometric support and a current mesh considered as a support.

If the support is of a *geometric nature*, `Geometry`, defined by a file, it gives the relationships between the vertices, boundary edges and boundary faces of the current mesh with

[1] In this way, it will be possible to add some fields that are not yet defined.

the geometric entities. Thus, a mesh vertex can be identical to a geometric vertex, a mesh edge can have a geometric edge as support and, in three dimensions, a face (a triangle or a quadrilateral) can have a geometric face as support. These relationships allow to classify the entities of the current mesh with respect to their membership to an entity defining the domain geometry (this information will be useful in particular when constructing finite elements of order greater than one).

If the support is a *(usual) mesh* by itself, `MeshSupportOfVertices`, defined by a file, it gives the relationships between the current mesh and the above mesh. A vertex of the current mesh belongs to an entity[2] of the support mesh (this information may be relevant when interpolating or transporting a solution from one mesh to the other in an adaptive iterative process for instance).

Hence, in an iterative computational process, the support for the mesh at a given iteration step is the mesh of the previous step. In this way, we indicate that a vertex, i, of the current mesh

- is identical with a vertex of the support,
- lies on an edge of the support at abcissa u,
- falls within a triangle of the support, u, v being the coordinates in the reference element,

A vertex not in this "table" is considered as a free vertex. The relationships defined in this way enable the software to know the location of a vertex using the reference element related to the support entity including this vertex. To use the reference element to arrive to the current element, one must use one of the following relations according to the geometric type of the element

- for an edge with endpoints k_1 and k_2

$$x_i^j = (1-u)x_{k_1}^j + ux_{k_2}^j$$

- for a triangle with vertices $k_l, \quad l = 1, 3$

$$x_i^j = (1-u-v)x_{k_1}^j + ux_{k_2}^j + vx_{k_3}^j$$

- for a quadrilateral with vertices $k_l, \quad l = 1, 4$

$$x_i^j = (1-u)(1-v)x_{k_1}^j + u(1-v)x_{k_2}^j + uvx_{k_3}^j + (1-u)vx_{k_4}^j$$

Remark. These informations are naturally known by the mesh generation algorithm and relatively easy to obtain. Moreover, when simplicial elements are used, the barycentric coordinates are obvious to obtain and thus are not strictly required to be stored.

Crack definition is the purpose of three fields, `CrackedEdges`, `CrackedTriangles` and `CrackedQuadrangles`; we specify then that an edge (resp. a face) is identical in terms of geometry to another edge (resp. face).

[2]For a boundary element, a projection will be needed to obtain the sought location.

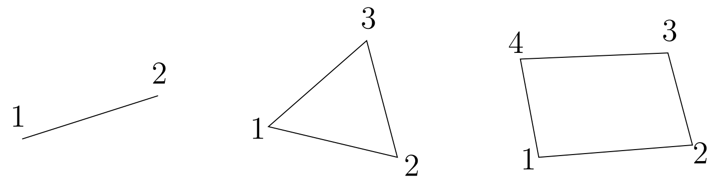


Figure 15: *Canonical numbering.*

The field, `EquivalencedEdges` indicate that two edges must be meshed the same way (e.g. the periodic meshes).

A comment about the meanning of the physical reference numbers is provided in the field `PhysicsReference`.

It is possible to include a file in the data structure, `IncludeFile`. This inclusion will be made without any compatibility insurance.

For some applications, it is useful to know the size of the domain, i.e. the extrema of its point coordinates, which is the meaning of the field `BoundingBox`.

4.1.1 A Geometric Data Structure

The general structure allows to specify a mesh describing the geometry of the given domain. This mesh is used both to define the *geometry* and the *physical conditions* of the problem under consideration.

In this case, some of the above fields are not relevant. We like to give some indications about this geometric structure in two dimensions. One is referred hereafter to see how such a structure can be effectively used to construct a geometric representation and to see how to use this representation for boundary meshing (remeshing) purpose.

We give the list of the fields used in this case by indicating at first the required fields

- `MeshVersionFormatted 0`
- `Dimension (I)` dim
- `Vertices (I)` NbOfVertices
  $\Big(\big(\texttt{(R)}\ x_i^j\,,\ \ j{=}1{,}\mathrm{dim} \big)\ ,\ \texttt{(I)}\ Ref\phi_i^v\,,\ \ i{=}1\,,\mathrm{NbOfVertices} \Big)$
- `Edges (I)` NbOfEdges
  $\big(\texttt{@@Vertex}_i^1\,, \texttt{@@Vertex}_i^2\,,\ \texttt{(I)}\ Ref\phi_i^e\,,\ \ i{=}1\,,\mathrm{NbOfEdges} \big)$

and then the optional fields

- `SubDomain (I)` NbOfSubDomain

  $$\Big(\ \texttt{(I)}\ type_i,\ \ \big\{\ type_i == 2:\ \ \texttt{@@Edge}_i\ \big\}\ \ ,\ \texttt{(I)}\ \mathrm{Orientation}_i\,,\ \texttt{(I)}\ Ref\phi_i^s\,,\ \ i{=}1\,,\mathrm{NbOfSubDomain} \Big)$$

- `Corners (I)` NbOfCorners
  $\big(\texttt{@@Vertex}_i\,,\ \ i{=}1\,,\mathrm{NbOfCorners} \big)$
- `RequiredVertices (I)` NbOfRequiredVertices
  $\big(\texttt{@@Vertex}_i\,,\ \ i{=}1\,,\mathrm{NbOfRequiredVertices} \big)$

- `RequiredEdges (I)` NbOfRequiredEdges
  $\Big($ `@@Edge`$_i$ **,** i=1 , NbOfRequiredEdges $\Big)$
- `TangentAtEdges (I)` NbOfTangentAtEdges
  $\Big($ `@@Edge`$_i$, `(I)` VertexInEdge , $\Big($ `(R)` x_i^j **,** j=1,dim $\Big)$ **,** i=1 , NbOfTangentAtEdges $\Big)$
- `AngleOfCornerBound (R)` θ
- `CrackedEdges (I)` NbOfCrackedEdges
  $\Big($ `@@Edge`$_i^1$, `@@Edge`$_i^2$ **,** i=1 , NbOfCrackedEdges $\Big)$
- `EquivalencedEdges (I)` NbOfEquivalencedEdges
  $\Big($ `@@Edge`$_i^1$, `@@Edge`$_i^2$ **,** i=1 , NbOfEquivalencedEdges $\Big)$
- `PhysicsReference (I)` NbOfPhysicsReference
  $\Big($ `(I)` $Ref\phi_i$, `(C*)` CommentOnThePhysic **,** i=1 , NbOfPhysicsReference $\Big)$
- `IncludeFile (C*)` filename
- `BoundingBox` $\Big($ `(R)` Min_i `(R)` Max_i **,** i=1 , dim $\Big)$

The geometric representation of a boundary in two dimensions will be detailed hereafter. At this time, just keep in mind that we will use the edges provided in the data structure so as to define some curves of order three in the following way [3]

- an edge whose endpoints are corners and if no additional information are provided will be represented by a straight segment,
- an edge whose endpoints are corners but whose tangent is provided at one endpoint will be represented by a curve of degree two,
- an edge whose endpoints are corners but whose tangents are provided at these corners will be represented by a curve of degree three,
- an edge whose endpoints are not corners and with no additional information will be represented by a curve of degree three. Indeed, we use in this case the adjacent edges so as to evaluate the tangents at the edge endpoints,
- etc.

In short, an edge defined by two informations will be approached by a straight line, three information allow to obtain a curve of degree two and four data allow for a approximation[4] of degree three.

This way of constructing the geometric support from a geometric support mesh being briefly established, we like to give an example.

We consider the rather simple domain of the Fig. 16 where Γ denotes (with no more precise information at this stage) the boundary part associated with the segments CF and FD. We aim at discussing the different ways to construct a mesh data structure by observing in each case the resulting geometric definition.

Hence, if we provide as data the following information

[3] A different choice is clearly possible leading to modify the type of representation and thus the way in which the related information are used furthermore.

[4] One also may say that an approximation of degree three is constructed with, in some cases, when some data are missing, the tangents defined by the edge itself.

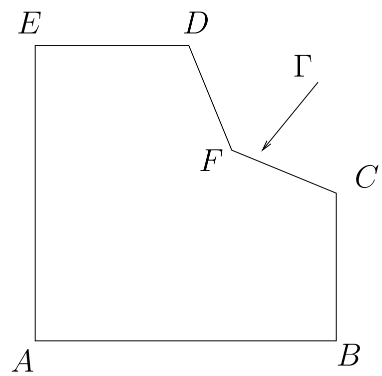


Figure 16: *The domain to be defined.*

- `MeshVersionFormatted 0`
- `Dimension` 2
- `Vertices` 6
  $(x_A, y_A, 1)$, $(x_B, y_B, 1)$, $(x_C, y_C, 1)$, $(x_D, y_D, 1)$, $(x_E, y_E, 1)$, $(x_F, y_F, 1)$
- `Edges` 6
  $(A, B, 1)$, $(B, C, 1)$, $(A, E, 1)$, $(E, D, 1)$, $(F, D, 1)$, $(F, C, 1)$
- `SubDomain` 1
  2 4 -1 10
- `Corners` 5
  A B C D E

Six points have been implicitely defined[5], as well as six edges, the domain is on the "right side" of the edge number 4, alias ED, its material number is 10. The edges AB, BC, AE and ED will be approximated by some straight lines, the "curve" Γ is the union of the segments FD and FC with, in F, a tangent defined thanks to DC (as F is not a corner) and, in D (resp. in C), a tangent supported by FD (resp. FC) as D (resp. C) is a corner, thus Γ will be a piecewise curve of degree two.

Remark. If the edge number 4 is DE (instead of ED), the sub-domain must be described by the sequence 2 4 1 10 instead of 2 4 -1 10.

How to define the geometric data structure so as Γ be a circle (at least a curve close enough to a circle)? It is only needed to enrich the structure by providing more points along Γ, then an approximation of degree two will be obtained for each terminal sub-segment and of a degree three elsewhere. One may also define the tangents at the endpoints so as to obtained an approximation of degree three everywhere.

One may notice that we do not explain how to construct the desired data structure from a practical point of view. Clearly a preprocessor (of CAD type) will be in charge of this task.

[5]In this example, a point and its number stand for the same notion, for instance the point A is the same as the point with number A.

5 The Two-Dimensional Case Mesh Data Structure

The mesh data structure, output of a mesh generation algorithm, refers to the geometric data structure and in some case to another mesh data structure.

In this case, the fields are

- `MeshVersionFormatted 0`
- `Dimension (I)` dim
- `Vertices (I)` NbOfVertices
  $\Big(\big(\texttt{(R)}\ \mathrm{x}_i^j\,,\ \mathrm{j{=}1,dim} \big)\,,\ \texttt{(I)}\ Ref\phi_i^v\,,\ \mathrm{i{=}1\,,NbOfVertices} \Big)$
- `Edges (I)` NbOfEdges
  $\Big(\texttt{@@Vertex}_i^1\,, \texttt{@@Vertex}_i^2\,,\ \texttt{(I)}\ Ref\phi_i^e\,,\ \mathrm{i{=}1\,,NbOfEdges} \Big)$
- `Triangles (I)` NbOfTriangles
  $\Big(\big(\texttt{@@Vertex}_i^j\,,\ \mathrm{j{=}1,3} \big)\,,\ \texttt{(I)}\ Ref\phi_i^t\,,\ \mathrm{i{=}1\,,NbOfTriangles} \Big)$
- `Quadrilaterals (I)` NbOfQuadrilaterals
  $\Big(\big(\texttt{@@Vertex}_i^j\,,\ \mathrm{j{=}1,4} \big)\,,\ \texttt{(I)}\ Ref\phi_i^t\,,\ \mathrm{i{=}1\,,NbOfQuadrilaterals} \Big)$
- `Geometry`
  `(C*)` FileNameOfGeometricSupport
  - `VertexOnGeometricVertex`
    `(I)` NbOfVertexOnGeometricVertex
    $\Big(\texttt{@@Vertex}_i\,, \texttt{@@Vertex}_i^{geo}\,,\ \mathrm{i{=}1,NbOfVertexOnGeometricVertex} \Big)$
  - `EdgeOnGeometricEdge`
    `(I)` NbOfEdgeOnGeometricEdge
    $\Big(\texttt{@@Edge}_i\,, \texttt{@@Edge}_i^{geo}\,,\ \mathrm{i{=}1,NbOfEdgeOnGeometricEdge} \Big)$
- `CrackedEdges (I)` NbOfCrackedEdges
  $\Big(\texttt{@@Edge}_i^1\,, \texttt{@@Edge}_i^2\,,\ \mathrm{i{=}1\,,NbOfCrackedEdges} \Big)$

When the current mesh refers to a previous mesh, we have in addition

- `MeshSupportOfVertices`
  `(C*)` FileNameOfMeshSupport
  - `VertexOnSupportVertex`
    `(I)` NbOfVertexOnSupportVertex
    $\Big(\texttt{@@Vertex}_i\,, \texttt{@@Vertex}_i^{supp}\,,\ \mathrm{i{=}1,NbOfVertexOnSupportVertex} \Big)$
  - `VertexOnSupportEdge`
    `(I)` NbOfVertexOnSupportEdge
    $\Big(\texttt{@@Vertex}_i\,, \texttt{@@Edge}_i^{supp}\,,\ \texttt{(R)}\ u_i^{supp}\,,\ \mathrm{i{=}1,NbOfVertexOnSupportEdge} \Big)$
  - `VertexOnSupportTriangle`
    `(I)` NbOfVertexOnSupportTriangle
    $\Big(\texttt{@@Vertex}_i\,, \texttt{@@Tria}_i^{supp}\,,\ \texttt{(R)}\ u_i^{supp}\,,\ \texttt{(R)}\ v_i^{supp}\,,\ \mathrm{i{=}1\,,NbOfVertexOnSupportTriangle} \Big)$
  - `VertexOnSupportQuadrilaterals`
    `(I)` NbOfVertexOnSupportQuadrilaterals
    $\Big(\texttt{@@Vertex}_i\,, \texttt{@@Quad}_i^{supp}\,,\ \texttt{(R)}\ u_i^{supp}\,,\ \texttt{(R)}\ v_i^{supp}\,,\ \mathrm{i{=}1\,,NbOfVertexOnSupportQuadrilaterals} \Big)$

5.1 bb File type for Store Solutions

The file is formatted such that:

`2 nbsol nbv 2`

$((\mathtt{U}_{ij}, \quad \forall i \in \{1, ..., \mathtt{nbsol}\}), \quad \forall j \in \{1, ..., \mathtt{nbv}\})$

where

- `nbsol` is a integer equal to the number of solutions.
- `nbv` is a integer equal to the number of vertex .
- $\mathtt{U}_{ij}$ is a real equal the value of the i solution at vertex j on the associated mesh background if read file, generated if write file.

5.2 BB File Type for Store Solutions

The file is formatted such that:

$\mathtt{2} \quad \mathtt{n} \quad \mathtt{typesol}^{\mathtt{1}} \quad ... \quad \mathtt{typesol}^{\mathtt{n}} \quad \mathtt{nbv} \quad \mathtt{2}$

$\left(\left(\left(\mathtt{U}_{ij}^{k}, \quad \forall i \in \{1, ..., \mathtt{typesol}^{k}\}\right), \quad \forall k \in \{1, ...\mathtt{n}\}\right) \quad \forall j \in \{1, ..., \mathtt{nbv}\}\right)$

where

- `n` is a integer equal to the number of solutions
- $\mathtt{typesol}^{\mathtt{k}}$, type of the solution number k, is
  - $\mathtt{typesol}^{\mathtt{k}} = \mathtt{1}$ the solution `k` is scalare (1 value per vertex)
  - $\mathtt{typesol}^{\mathtt{k}} = \mathtt{2}$ the solution `k` is vectorial (2 values per unknown)
  - $\mathtt{typesol}^{\mathtt{k}} = \mathtt{3}$ the solution `k` is a 2×2 symmetric matrix (3 values per vertex)
  - $\mathtt{typesol}^{\mathtt{k}} = \mathtt{4}$ the solution `k` is a 2×2 matrix (4 values per vertex)
- `nbv` is a integer equal to the number of vertices
- $\mathtt{U}_{ij}^{k}$ is a real equal the value of the component i of the solution k at vertex j on the associated mesh background if read file, generated if write file.

5.3 Metric File

A metric file can be of two types, isotropic or anisotropic.

the isotrope file is such that

`nbv 1`

$\mathtt{h}_i \quad \forall i \in \{1, ..., \mathtt{nbv}\}$

where

- `nbv` is a integer equal to the number of vertices.
- $\mathtt{h}_i$ is the wanted mesh size near the vertex i on background mesh, the metric is $\mathcal{M}_i = h_i^{-2} Id$, where Id is the identity matrix.

The metric anisotrope

`nbv 3`

$\mathtt{a11}_i, \mathtt{a21}_i, \mathtt{a22}_i \quad \forall i \in \{1, ..., \mathtt{nbv}\}$

where

- `nbv` is a integer equal to the number of vertices,
- $\mathrm{a11}_i$, $\mathrm{a12}_i$, $\mathrm{a22}_i$ is metric $\mathcal{M}_i = \left(\begin{smallmatrix} a11_i & a12_i \\ a12_i & a22_i \end{smallmatrix}\right)$ which define the wanted mesh size in a vicinity of the vertex i such that h in direction $u \in \mathbb{R}^2$ is equal to $|u|/\sqrt{u \cdot \mathcal{M}_i\, u}$, where $\cdot$ is the dot product in $\mathbb{R}^2$, and $|\cdot|$ is the classical norm.

5.4 List of AM_FMT, AMDBA Meshes

The mesh is only composed of triangles and can be defined with the help of the following two integers and four arrays:

`nbt` is the number of triangles.

`nbv` is the number of vertices.

`nu(1:3,1:nbt)` is an integer array giving the three vertex numbers counterclockwise for each triangle.

`c(1:2,nbv)` is a real array giving the two coordinates of each vertex.

`refs(nbv)` is an integer array giving the reference numbers of the vertices.

`reft(nbv)` is an integer array giving the reference numbers of the triangles.

AM_FMT Files In fortran the `am_fmt` files are read as follows:

```
open(1,file='xxx.am_fmt',form='formatted',status='old')
  read (1,*) nbv,nbt
  read (1,*)  ((nu(i,j),i=1,3),j=1,nbt)
  read (1,*)  ((c(i,j),i=1,2),j=1,nbv)
  read (1,*)  ( reft(i),i=1,nbt)
  read (1,*)  ( refs(i),i=1,nbv)
close(1)
```

AM Files In fortran the `am` files are read as follows:

```
open(1,file='xxx.am',form='unformatted',status='old')
  read (1,*) nbv,nbt
  read (1)  ((nu(i,j),i=1,3),j=1,nbt),
&   ((c(i,j),i=1,2),j=1,nbv),
&   ( reft(i),i=1,nbt),
&   ( refs(i),i=1,nbv)
close(1)
```

AMDBA Files In fortran the `amdba` files are read as follows:

```
open(1,file='xxx.amdba',form='formatted',status='old')
  read (1,*) nbv,nbt
  read (1,*) (k,(c(i,k),i=1,2),refs(k),j=1,nbv)
  read (1,*) (k,(nu(i,k),i=1,3),reft(k),j=1,nbt)
close(1)
```

msh Files In fortran the `msh` files are read as follows:

```
open(1,file='xxx.msh',form='formatted',status='old')
  read (1,*) nbv,nbt
  read (1,*) ((c(i,k),i=1,2),refs(k),j=1,nbv)
  read (1,*) ((nu(i,k),i=1,3),reft(k),j=1,nbt)
close(1)
```

ftq Files In fortran the `ftq` files are read as follows:

```
open(1,file='xxx.ftq',form='formatted',status='old')
 read (1,*) nbv,nbe,nbt,nbq
 read (1,*) (k(j),(nu(i,j),i=1,k(j)),reft(j),j=1,nbe)
 read (1,*) ((c(i,k),i=1,2),refs(k),j=1,nbv)
close(1)
```

where if `k(j) = 3` then the element j is a triangle and if `k = 4` the the element j is a quadrilateral.

6 Graphical interface

If the software was compiled with the FLAGS= -DDRAWING, then we have a graphic interface:

The inputs for the graphic are: (case sensitive)

`=`	restore the default view points
`r`	refresh the screen
`+`	zoom +
`-`	zoom -
`T`	draw the sub-domain number
`g`	draw the geometry
`m`	draw the metric ellipse
`V`	draw all the vertices
`k`	draw the triangle contening the mouse point or the nearest edge (if outside point)
`v`	find the nearest vertex to the mouse point
`s`	find the nearest vertex to the mouse point and draw the metric
`o`	find the nearest vertex to the mouse point and try to optimize
`t`	find the triangle contening the mouse point (stupid way) for debug
`c`	draw the center of inscribe circle of the triangle in the metric
`e`	find the nearest edge in a triangle contening the mouse point (stupid way) for debug
`f`	continue to the next step
`q`	make an error and do not continue
`B`	activate the background mesh by adding a step, to desactive enter `f`

7 Related Tools

The command `cvmsh2 filein fileout [ -g filegeom ] [ -thetamax AngleOfCornerBound` transform the `filenin` in BD mesh store in `fileout` plus a geometry store in `filegeom` if `-g filegeom` arguments are given.

The filename is composed `prefix.suffix`, the type of the input file is `am`, `am_fmt`, or `amdba` and is given by the suffix part.

8 BUGS and Limitations

The BD mesh files conteins generaly a link to other BD mesh file, like geometry file or background mesh file. So if you move some mesh file don't forget to move also all the related file. If you want to use/exec in an other directory, you have to put absolute pathname when you construct all files.

If you make an adapation loop with no BD mesh file, the geometry was reconstruct a each step, so you can lose something (the geometry).