INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

EMC2 Wysiwyg 2D finite elements mesh generator

Eric Saltel, Frédéric Hecht

N° 118

Octobre 1995

PROGRAMME 6

Rapport technique

ISSN 0249-6399

EMC2 Wysiwyg 2D finite elements mesh generator

Eric Saltel, Frédéric Hecht *

Programme 6 — Calcul scientifique, modélisation et logiciel numérique
Projet MENUSIN

Rapport technique n° 118 — Octobre 1995 — 67 pages

Abstract: Emc^2 is a portable, interactive and graphic software Edition of two dimensional geometry and mesh. We can create and modify the geometry (CAD), define the discretization on the lines, define the subdomains, and define some reference numbers to take into account the boundary conditions and material properties. The elements of the mesh are traingles and quadrilaterals. We have two kind of meshes: grid mesh and Delaunay Voronoï(automatic mesh). We make the edition of the mesh by moving, removing, adding vertices, by regularization, or by transformations (symmetry, rotation,..), etc.

Key-words: MESH, GRID, FINITE ELEMENT, CAD, GEOMETRIE, TRIANGLE,QUADRANGLE

(Résumé : tsvp)

*Translated by C. Le Dret

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : (33 1) 39 63 55 11 – Télécopie : (33 1) 39 63 53 30

Un logiciel d'édition de maillages et de contours bidimensionnels

Résumé : Emc^2 est un logiciel portable, graphique et intéractif d'édition de maillage et contours en 2 dimensions. Il permet de générer intéractivement des maillages bidimensionnels pour la méthode des éléments finis en définissant la géometrie (D.A.O), la discrétisation des contours, les sous-domaines et les numéros de reférence (afin d'introduire un lien avec la physique: conditions aux limites, propriétés des matériaux). Ces maillages, formés de triangles ou de quadrangles, sont de type grille ou de type Delaunay-Voronoï. Il est possible d'éditer un maillage en ajoutant, supprimant, déplaçant des sommets,... et en lui appliquant des transformations affines: symetrie, rotation,...) etc.

Mots-clé : MAILLAGE, ELEMENTS FINI, D.A.O, GEOMETRY, TRIANGLE, QUADRANGLE

Contents

List of Figures

Chapter 1

INTRODUCTION

Une des difficultés de la méthode des éléments finis est la définition du domaine de calcul Ω et la construction d'un maillage de Ω. Etant donné que l'on est amené à résoudre des problèmes de plus en proches de la réalité, la complexité des domaines de calcul croît parallèlement,et le temps humain passé à définir la géométrie et générer les maillages peut devenir exorbitant si l'on ne dispose pas d'outils adéquats.

Ces différentes considérations ainsi que le développement des stations de travail (graphiques) ont motivé le développement et la réalisation du logiciel Emc^2 (pour **E**dition de **m**aillages et de **c**ontours en **2** dimensions).

L'utilisation de ce logiciel portable, graphique, rapide, basé sur des menus (*qui orientent les actions pour la définition des géométries et le maillage de celles-ci*), diminuera de façon significative le temps nécessaire à la préparation des données dans la méthode des éléments finis.

Chapter 2

General remarks

The program is divided into three main, mutually exclusive, applications:

1. the **CONSTRUCTION** application: used for editing and creating the geometry (contours);
2. the **PREP_MESH** application: used for editing and defining the mesh for the contours, the references of the lines, points and subdomains;
3. the **EDIT_MESH** application: used for editing and creating a triangular or quadrangular mesh.

In each of the applications, we can also have access to the following menus:

- *GENERAL* used among other things for moving from one application to another, for saving or restoring a database, for making hard and soft screen copies, for reexecuting a run or for quitting Emc^2...
- *GRAPHIC SCREEN MANAGEMENT* used for handling database display,
- *CALCULATOR* used for entering numerical data, which can be the result of an expression, whenever necessary, or to display certain values such as the radius of a circle or an arc of a circle, the length of an arc or of a segment...
- *SELECT* used for defining the type of element that we wish to select, for example, coordinate, point, line, circle, arc, segment, spline... This menu is application dependent since the ITEMs to be selected vary according to the application.

Each main application has a specific menu of its own. All applications are driven by LL1 grammars.

The screen (figure 2.1) is divided into 9 zones: 5 are menus ((1), (2), (4), (7), (6)), 3 are display zones ((8), (9), (5)) , the center zone is for graphics.

The **GENERAL**, **CALCULATOR**, **SCREEN MANAGEMENT** menus are fixed and application independent. The **APPLICATION** and **SELECT** menus obviously depend on the current application.

The program is interactive, all entries are made with the mouse and keyboard. The mouse is used to select menu ITEMS, as well as graphic elements. Graphic elements can only be selected after the desired type of element has been chosen in the **SELECT** menu. The keyboard is used to enter text, for example file names, as well as *short cuts* (ITEMS of certain menus have keyboard equivalents).

2.1 The CONSTRUCTION application

This application is used to define the geometric contours of the domain with the help of points, segments, arcs of circles and splines. Additionnally, two other construction entities are useful in certain cases: straight lines and circles. These last two entities are only used in the **CONSTRUCTION** application.

All these entities, except splines, can be constructed by using elementary geometry theorems. When there are multiple solutions, ambiguities are removed by using the following heuristic method: the selected points are close to the points of tangency of the elements (point, line, circle, arc, segment).

A spline is a C^1-curve passing through a series of points. It is closed if the first and last points coincide.

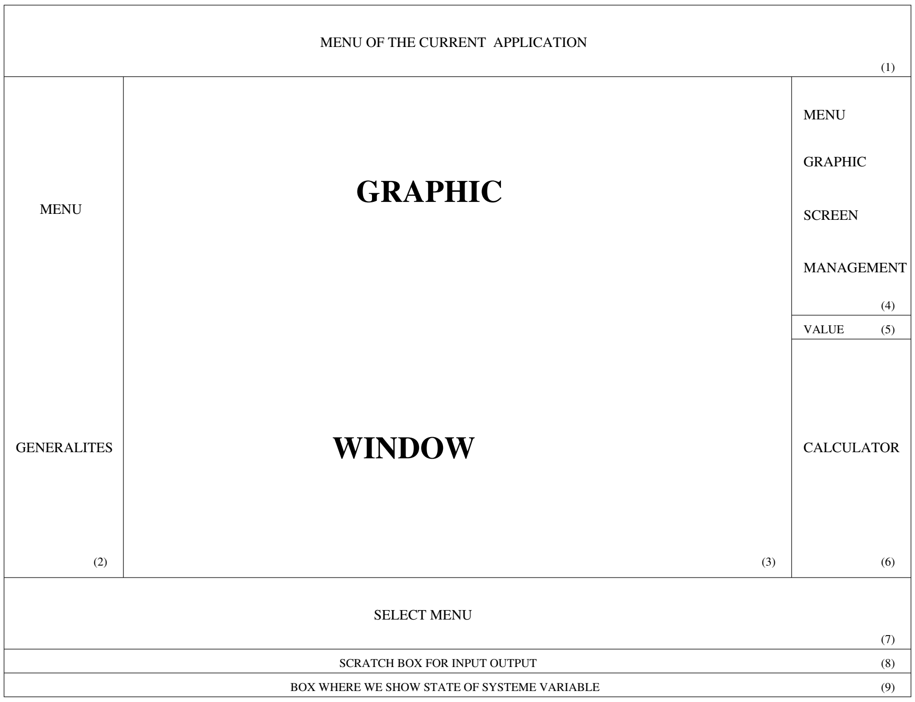

Figure 2.1: Depiction of the screen

All these entities can be duplicated by using the following affine transformations: symmetry, rotation, homothety, translation.

Moreover, it is possible to:

- round off angles,
- divide segments, arcs, splines with points, segments, arcs, splines,
- destruct entities using the **GENERAL** menu.

2.2 The PREP_MESH application

At this point the connex components of the boundaries of the subdomains of Ω are known. We will call these connex components simply components.

This application is used to define the discretization of the entities defining the boundaries of the domain and subdomains (boundary of materials). It is also used to define the reference numbers of the subdomains, lines and points, with the aim of entering various physical data, for example, several boundary conditions, several materials.

This application is also used to create a data file so as to interface with the MODULEF mesh generator APNOXX, by defining the subdomains as a list of components where the first component is the exterior component—the other components being the components of the holes—plus internal lines and internal points to be enforced.

2.3 The EDIT_MESH application

When moving to this application, it is possible to modify the default value of certain parameters of the mesh generator. The program constructs a triangular or grid quadrangular mesh of the subdomains defined in the previous step, or of all subdomains if none has been defined.

At the point, we can edit the mesh by:

- adding internal vertices to subdomains
- suppressing vertices or subdomains
- swapping internal edges of a quadrilateral
- moving vertices
- regularizing the mesh or making it *Delaunay*
- quadrangulating the mesh
- transforming subdomains with symmetries, rotations, homotheties, translations
- modifying the references of vertices, edges and subdomains (regions)
- saving or restoring a mesh under different formats: MESH, the Emc^2 internal structure; the MODULEF NOPO structure; a simplified structure and so on
- cracking lines of the mesh: the points situated on each side of the lines are duplicated
- renumbering so as to decrease the size of the profile of the P^1 or Q^1 finite element matrices.

2.4 A very simple detailed example

For a change, we are going to mesh a unit circle instead of a square. First, we must construct the circle, i.e. a 360° arc, then define the discretization of the arc and finally contruct the mesh. These three phases correspond respectively to the three applications CONSTRUCTION, PREP_MESH and EDIT_MESH. Here is the list of steps to be taken.

1. Run emc2, then enter the machine-dependent number of the graphic peripheral device. After initialization we are implicitly in the construction application and the selection mode is mouse point (MOUSE P). To construct the arc:

   (a) click the **ARC** item on top of the construction menu,

   (b) click the **CENTER** item of the same menu to define the center,

   (c) click the **XY POINT** item of the select menu,

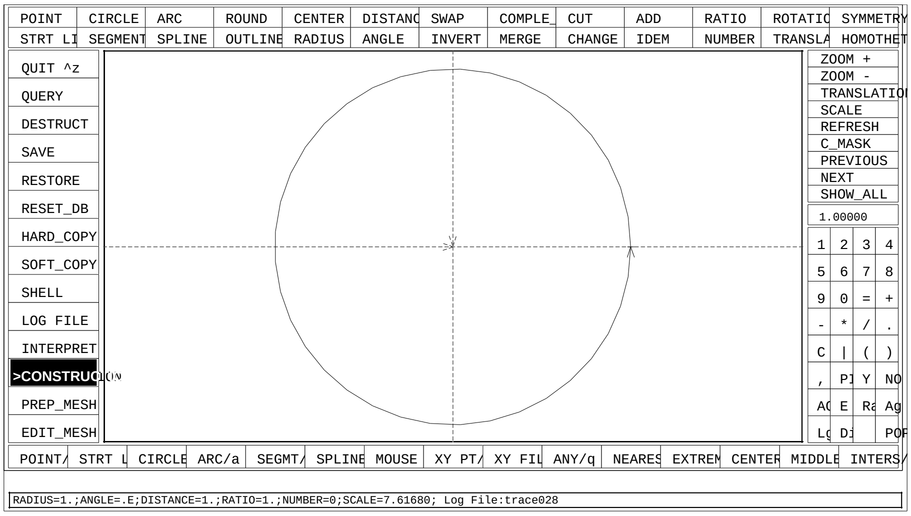

Figure 2.2: The unit circle

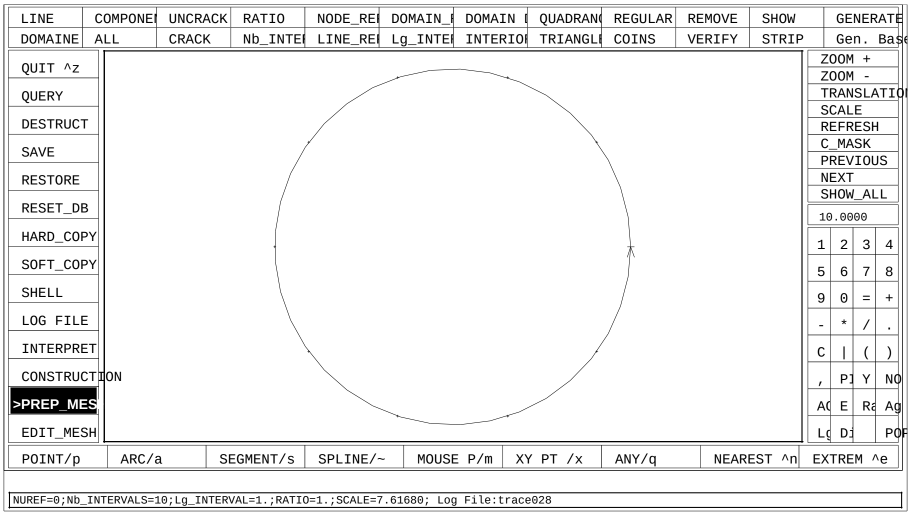

Figure 2.3: The discretized unit circle

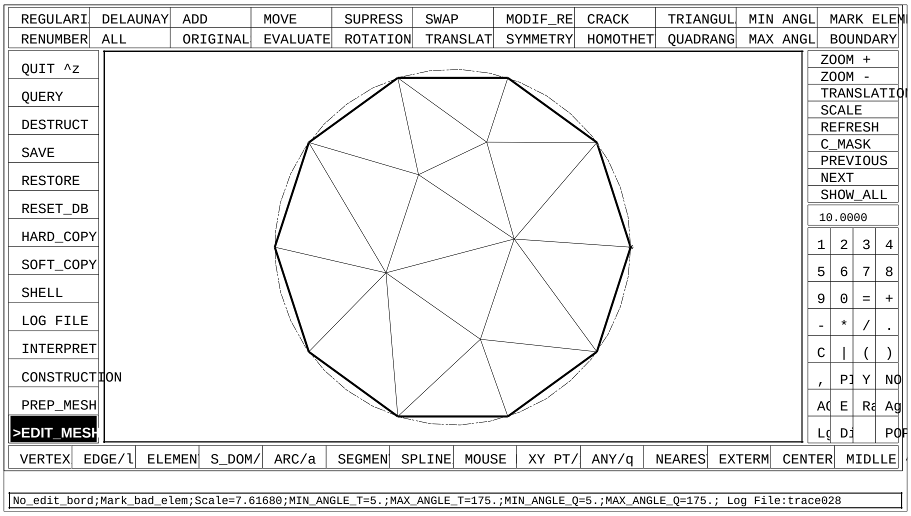

Figure 2.4: The meshed unit circle

(d) enter `0=0=` on the keyboard. This sets the center at point (0,0),

(e) click the `RADIUS` item on top of the select menu,

(f) enter 1= on the keyboard, which sets the radius to 1 (the radius state variable is set to 1 and displayed on the bottom of the screen).

a 360° arc of circle centered at (0,0) with a radius of 1 is displayed. *Note: although the angle state variable is equal to* 0*, the angle of the arc is actually* 360° *(* $0 \equiv 360 \, mod \, 360$*).*

2. The circle is small, around 2cm, because the implicit scale is equal to 1. To see it full screen (figure 2.2), click `SHOW_ALL` in the screen management menu (top right).

3. Now we have constructed the circle and we switch to the PREP_MESH application by clicking on the corresponding item of the general menu.

   To define the number of points in the arc:

   (a) click on `NB_INTERVAL`

   (b) enter the number of intervals `10=`

   (c) select the arc by clicking in the `ARC` item of the select menu (bottom)

   (d) click near the arc in the graphic window.

   The points generated appear on the screen (figure 2.3). We construct the mesh by moving to the EDIT_MESH application. For this, we click the `EDIT_MESH` item in the general menu, then type `(CR)`four times to use the default options in response to the four questions posed by the program. The meshed circle appears on the screen (figure 2.4). The mesh can be saved with the `SAVE` item of the general menu, by entering:

   (a) the type of mesh to be saved, for example by typing `am_fmt(CR)`

   (b) the name of the prefix of the mesh by typing `circle<return>`

   thus a file entitled *circle.am_fmt* containing the mesh is generated.

2.5 A less trivial example

Construction of a mesh between 8 circles tangent to the 3 dotted circles (cf. figure 2.5). The arcs are divided in the neighborhood of the points of "near tangency" for future mesh refinement. The different stages are shown in figures 2.5,2.6 and 2.7 and enlargements of the mesh in figures 2.8 and 2.9. Figure 2.9 shows a very strong zoom (around 360 times) around a point of near tangency.

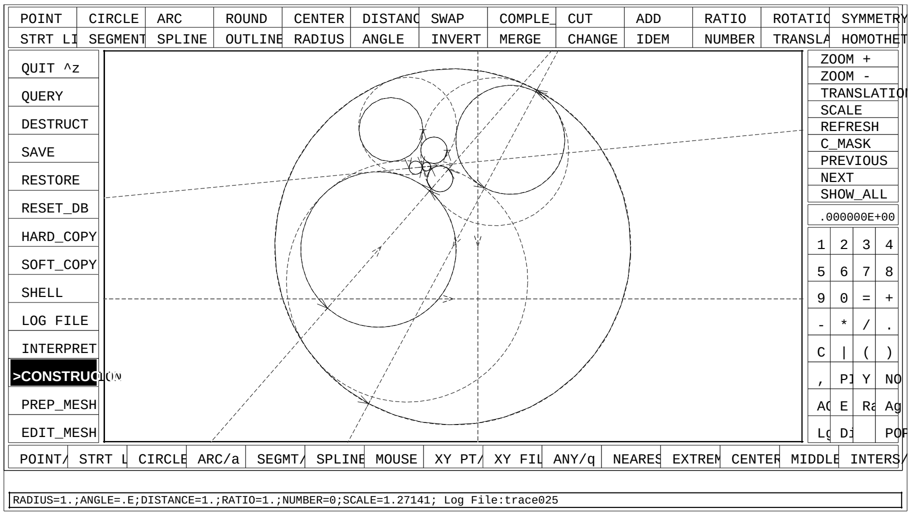

Figure 2.5: The geometry of the domain

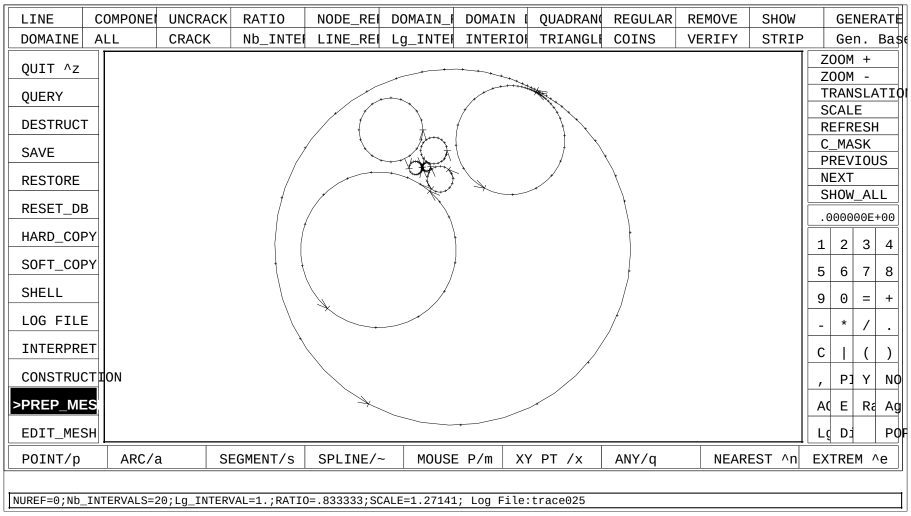

Figure 2.6: The discretization of contours.

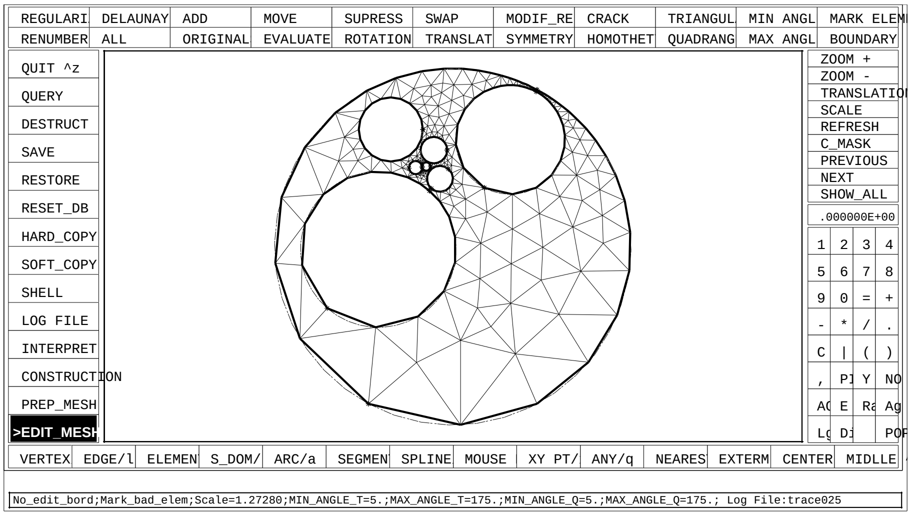

Figure 2.7: Display of the whole mesh.

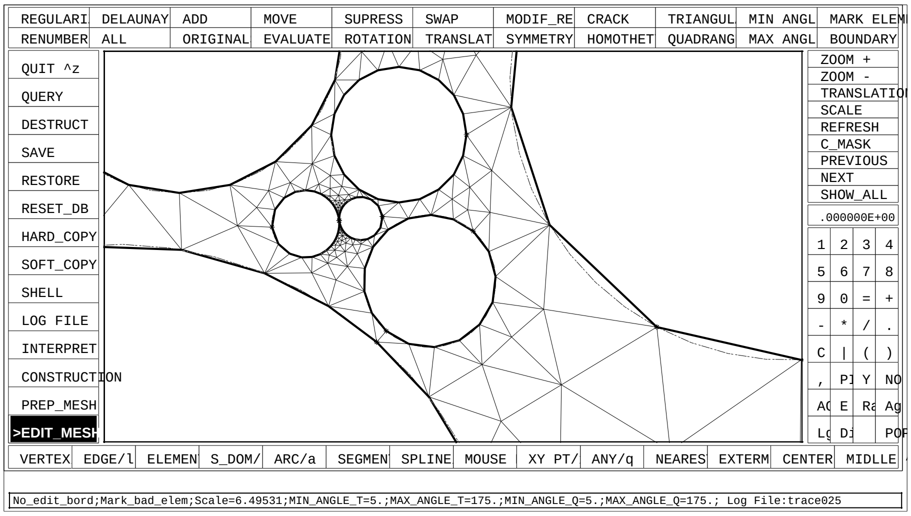

Figure 2.8: A first zoom.

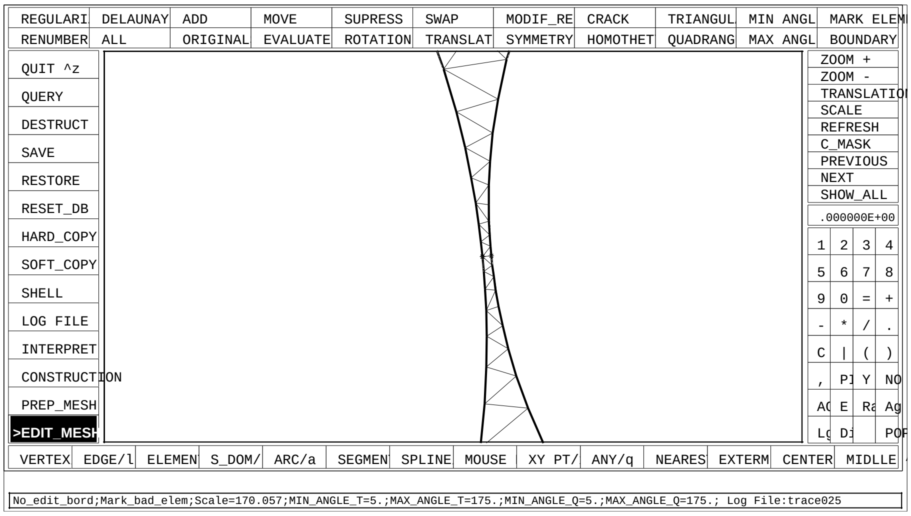

Figure 2.9: Zoom around one of the points of near tangency

Chapter 3

The global menus

Using the menus the user creates, edits, destructs and so on objects stored in two databases (DB), one for the geometry and the other for meshes.

3.1 Selection menus

These menus, shown at the bottom of the graphics window, make it possible to select the *type of graphics selection* we want (a point, a straight line, etc.). They depend on the application currently being used (cf. figures 3.1, 3.2, 3.3). It is absolutely essential to understand the outputs of these menus because all the rest depends on their functioning. All these outputs, for example, a circle, a number or the extremity of an arc, are marked (?) and the marks (or types ?) define the selection syntactical entities known by the grammars which manage the three applications.

POINT	STRT LINE	CIRCLE	ARC	SEGMENT	SPLINE	MOUSE P	XY POINT	→ → →

← ← ←	XY FILE	ANY	NEAREST	EXTREM	CENTER	MIDDLE	INTERS

Figure 3.1: The select menu for the CONSTRUCTION application

POINT	ARC	SEGMENT	SPLINE	MOUSE P	XY POINT	ANY	NEAREST	EXTREM

Figure 3.2: The select menu for the PREP_MESH application

The constructable types are :

- `<COORD>` current point (mouse point, XY Point or extremity, center, middle of an element of the DB)
- `<POINT>` point of the DB
- `<LINE>` straight line of the DB
- `<CIRCLE>` circle of the DB
- `<ARC>` arc of the DB
- `<SEGMENT>` segment of the DB

VERTEX	EDGE	ELEMENT	S_DOM	ARC	SEGMENT	SPLINE	MOUSE P	→ → →

← ← ←	XY POINT	ANY	NEAREST	EXTREM	CENTER	MIDDLE

Figure 3.3: The select menu for the EDIT_MESH application

- `<SPLINE>` spline of the DB
- `<VALUE>` numerical value (entered with the calculator)
- `<VERTEX>` vertex of the mesh
- `<EDGE>` edge of the mesh
- `<ELEMENT>` finite element (triangle or quadrangle) of the mesh
- `<S_DOM>` subdomain of the mesh (connected component of the mesh)

These menus depend on the application being used as they define the type of element to be selected. The menu ITEMs (their keyboard shortcut is in parenthesis) can be classified in several categories.

1. the items `POINT (p)`, `STRT LINE (l)`, `CIRCLE (c)`, `ARC (a)`, `SEGMENT (s)`, `SPLINE (~)` are used to select the element of the DB closest to the clicked point, the type of selection depends on the kind of element found and by the item in categories (4), (5) that is used. If there is an incompatibility, the item in category (4) is forced to change to(?) `NEAREST`.
2. The items `MOUSE P (m), XY POINT (x), XY FILE (<)` are used to enter current points respectively as follows:
   - points can be defined by clicking the mouse in the graphic window;
   - points can be defined by a list of couples of values (coordinates) entered with the caculator;
   - points can be read from a 'FORMATTED' file containing 2 coordinates (real) per line up to the end of the file (EOF), the name of the file is requested in the scratch zone;

   Note: in the last two cases the point created can be outside the graphic window;
3. the `ANY (q)` item has no effect on the selection: the element of the DB closest to the clicked point is selected. The type of selection depends on the kind element found and by the item in categories 4, 5 that is used.
4. the `NEAREST` (`(ctrl)`)n)item is used to effectively select an element of the DB and the type of selection is that of the element.
5. the `EXTREM(ITY) ((ctrl))e), CENTER ((esc)c), MIDDLE ((ctrl))m)` items are used to define what we are going to select in the element of the DB (cf. (1) and (3)), i.e., we select the point (selection type `<COORD>`) that is respectively the extremity, the center or the middle of the selected DB element, if this makes sense. Otherwise, the main mode of selection is forced to change to ANY, cf. 3. while keeping the item in category 5.
6. the INTERSECTION(i) item is used to select the current point which is the intersection of 2 elements of the DB selected in the next paragraph (the selection type generated is `<COORD>`).
7. the `ELEMENT(f), EDGE(d), VERTEX(v), S_DOM(r)` items are used to select respectively the element—either triangle or quadrangle—containing the clicked point, this is for the selection type `<ELEMENT>`, the edge in an element closest to the clicked point, this is for the selection type `<EDGE>`, , the vertex in an element closest to the clicked point, this is for the selection type `<VERTEX>` or the subdomain containing the clicked point, this is for the selection type `<S_DOM>`.

Important note: In effect, the selection only becomes effective after we have clicked on an element in the graphic window. It is possible to click in the menu items as many times as we want so as to obtain a combination of marked items. A selection defines the type of selection and it preserves the cursor coordinates at the instant of the click. These coordinates are often useful for the application being used to remove semantic ambiguities or choose between multiple solutions to the problem being treated. For example,

- if we want to construct a circle tangent to 3 circles, there are generally 8 solutions. The program will choose the solution such that the points of tangency are closest to the selection points,

- the intersection of two circles gives two points. In INTERSECTION, the ambiguity is removed by retaining the point closest to the one used to select the second circle.

If we try to select something that does not exist, we get the message *you have selected nothing*, which has no effect whatsoever.

The selections possible depend on the application since all types of selection are not always needed. For example, circles and straight lines are only used in the CONSTRUCTION application, elements, edges, vertices and subdomains only exist in the EDIT_MESH application.

3.1.1 Shortcuts

Keyboard shortcuts that correspond to ITEMs of categories 1,2,3,4,7 are equivalent to 2 clicks, the first in the ITEM box and the following on the cursor point if it is in the graphic window. Thus the mode of selection has now changed and it is the element closest to the cursor which is selected.

3.1.2 Practical utilization

An element is selected by clicking next to it (the program searches for the element closest to the click). When we click a menu, it is marked with a > sign in front of the text of the menu. In the selection menus, two boxes can be marked at the same time according to the following combinations:

```
item      (shortcut)  ( possible combinations)
ANY       (q)         ( NEAREST, EXTREMITY, CENTER, MIDDLE )
POINT     (p)         ( NEAREST)
STRT LINE (l)         ( NEAREST)
CIRCLE    (c)         ( NEAREST, CENTER )
ARC       (a)         ( NEAREST, EXTREMITY, CENTER, MIDDLE )
SEGMENT   (s)         ( NEAREST, EXTREMITY, MIDDLE )
SPLINE    ( )         ( NEAREST, EXTREMITY )
MOUSE P   (m)         ( NEAREST)
XY POINT  (x)         ( NEAREST)
VERTEX    (v)         ( NEAREST)
EDGE      (d)         ( NEAREST)
ELEMENT   (f)         ( NEAREST)
S_DOM     (r)         ( NEAREST)
```

Other combinations are not valid. A selection is not effective except when an element on the screen has been clicked. In fact the boxes may be clicked as many times as necessary to obtain the desired combination.

3.2 The calculator menu

This menu (cf. figure 3.4) can also be considered as a selection menu. It can be used to define a numerical value which is of type `<VALUE>`.

We can write any FORTRAN expression. The effective computation is achieved by clicking on the = box (shortcut =) or by typing (CR).

The ITEMs with their keyboard shorcut between parentheses are:

1	2	3	4
5	6	7	8
9	0	=	+
−	∗	/	.
C	|	(	)
,	*PI*	*Y*	*NO*
AC	*E*	*Ra*	*Ag*
Lg	*Di*		*POP*

We move from one menu to the other by clicking the POP box

SIN	*COS*
TAN	*ATAN*
*ATAN*2	*EXP*
LOG	*LOG*10
SQRT	*MOD*
ABS	*SIGN*
INT	*NINT*
MIN	*MAX*
	POP

Figure 3.4: The two calculator menus

1	(1)	numbers
2	(2)	
3	(3)	
4	(4)	
5	(5)	
6	(6)	
7	(7)	
8	(8)	
9	(9)	
.	(.)	decimal point
+	(+)	
-	(-)	
∗	(∗)	
/	(/)	
)	())	
(	(()	
,	(,)	comma, separator for function arguments
=	(=)	
|	(|)	exponentiation operator
PI		$= \pi$
Y		$= 0$
NO		$= 1$
E	(E)	exponent for numbers in exponential format
C	(Z)	resets the calculator accumulator
AC	(R)	resets the calculator
Ra		= radius of a selected element (`<ARC>` or `<CIRCLE>`)
Ag		= angle of a selected element (`<ARC>`)
		or angle between two `<STRT LINE>`
Lg		= length of a selected element (`<ARC>` or `<SEGMENT>`
		or `<SPLINE>`).
Di		= distance between a `<POINT>` and another `<POINT>`, a `<STRT LINE>`,
		a `<CIRCLE>` or an `<ARC>`;
		or between two parallel `<STRT LINE>`,
		or distance between two `<CIRCLE>`.
POP		used to display the calulator menu of FORTRAN functions
		SIN, COS, TAN, ASIN, ACOS, ATAN, ATAN2, EXP, LOG,
		LOG10, INT, NINT, MOD.

Examples: to enter

1.234567E9 type `1.234567E9(CR)`

0.5 type `1/2=`

3.14159... click the `PI` box and type `(CR)`

2×circle radius type `2*`, click the `Ra` box, then select the `<circle>` (by clicking the circle box of the selection menu then clicking near the circle in the graphic window) and finally type `(CR)`.

3.3 The general menu

This menu (cf. figure 3.5) is used to perform general operations: move between CONSTRUCTION, PREP_MESH and EDIT_MESH, save, restore, screen copy or execute a system command. To initiate an operation, click in the corresponding item.

QUIT
QUERY
DESTRUCT
SAVE
RESTORE
RESET_DB
HARD_COPY
SOFT_COPY
SHELL
LOG FILE
INTERPRET
CONSTRUCTION
PREP_MESH
EDIT_MESH

Figure 3.5: The general menu

The operations of the general menu are:

3.3.1 Exiting the program

`QUIT`

Terminates the execution of the program.

3.3.2 Querying the database

`QUERY`

Queries the selected elements. Their characteristics are displayed in the input-output or scratch zone.

3.3.3 Destructing elements of the database

`DESTRUCT`

Only available in the CONSTRUCTION application. Destructs selected elements of the database. Warning: if we destruct a POINT, the program checks if this POINT is used to define a SPLINE. If this is the case, it is also removed from the SPLINE.

3.3.4 Saving the database

`SAVE`

Saves the DB in a file. The name of the file is composed a prefix and a suffix. The system asks for the prefix in the scratch zone (figure 2.1). According to the case, the suffix can be `.emc2_bd, .mesh, .nopo, .am, .am_fmt, .amdba` and it depends on the current application:

- In the CONSTRUCTION application, construction elements are saved together with their reference number (points, straight lines, circles, arcs, segments and splines). The state of the system is saved as well. The suffix is `.emc2_bd`.
- In the PREP_MESH application, SAVE works the same as in CONSTRUCTION. Additionally, the definition of the domains is saved. The suffix is `.emc2_bd`.
- In the EDIT_MESH application, different types of meshes can be saved. The type is entered in the scratch zone, before entering the file prefix. Implicitly known types are:

  mesh (default) The MESH_DB is saved with its references to the DB defining the geometry, i.e., the contours of the subdomains. The suffix is `.mesh`.

  nopo A Modulef NOPO data structure is extracted from the MESH_DB and written in a file with suffix `.nopo`. The curves defining the contours are lost.

  am An AM-type simplified structure—see appendix 9.4.1—is extracted from the MESH_DB and written in a nonformatted file with suffix `.am`. The curves defining the contours and the edge references are lost.

  am_fmt Same as am. However, the storage file is formatted with suffix `.am_fmt`, see appendix 9.4.2.

  amdba An AMDBA-type structure (see appendix 9.4.3) is extracted from the MESH_DB and written in a formatted file with suffix `.amdba`. The curves defining the contours and certain references are lost.

  Remark: It is possible to save meshes under other formats by modifying the `ecrmsh` subroutine.

3.3.5 Restoring the database

`RESTORE`

Restores the DB of a file. The name of the file is composed of a prefix and a suffix. The system asks for the prefix in the scratch zone (figure 2.1). According to the case, the suffix can be `.emc2_bd, .mesh, .nopo, .am, .am_fmt, .amdba`. The data restored depend on the current application.

- In the CONSTRUCTION application, construction elements are restored together with their reference number (points, straight lines, circles, arcs, segments and splines). The state of the system is restored as well. The suffix is `.emc2_bd`.
- In the PREP_MESH application, RESTORE works the same as in CONSTRUCTION. Additionally, the definition of the domains is saved. The suffix is `.emc2_bd`.
- In the EDIT_MESH application, different types of meshes can be restored. The type is entered in the scratch zone, before entering the file prefix. Implicitly known types are:

  mesh restores a mesh and its contours, whose geometrical definitions are saved. The suffix is `.mesh`.

  nopo The MESH_DB is constructed from a Modulef NOPO data structure. The curves defining the contours are reconstructed with segments and crack edition is lost. The suffix is `.nopo`.

  am The MESH_DB is constructed from an AM-type structure, see appendix 9.4.1. The curves defining the contours are reconstructed with segments. Crack and edge reference edition are lost. The suffix is `.am`.

  am_fmt Same as am. However, the file is formatted with suffix `.am_fmt`, see appendix 9.4.2.

  amdba The MESH_DB is constructed from an AMBDA-type structure, see appendix 9.4.3. The curves defining the contours are reconstructed with segments and crack edition is lost. The suffix is `.amdba`.

  Remark: It is possible to restore meshes saved under other formats by modifying the `addmsh` subroutine.

  Caution: The recently read mesh is not merged with already existing meshes.

3.3.6 Resetting

`DB_RESET`

Clears everything stored in the 2 databases.

3.3.7 Printing the screen

`HARD_COPY`

Makes a bitmap, screen resolution, hard copy of the screen. *Depends on the system and on the environment (cf.* `hardcp` *subroutine).*

3.3.8 Redrawing the window on another graphic peripheral device

`SOFT_COPY`

Displays the screen on another Fortran 3d peripheral device. *Depends on the system and on the environment (cf.* `softcp` *subroutine)*

3.3.9 Executing a system command

`SHELL`

Executes a system command (shell). The system asks for the the command(s) in a character string. *Depends on the system (cf.* `exec` *subroutine).*

3.3.10 Saving a run

`LOG FILE`

Saves all user operations in a file. The file name is composed of prefix given by the user and the suffix `.trace_emc2`. If the prefix starts with a + sign, it is appended to the trace file. A default trace file is open. Its prefix is `tracexxxx`, where xxxx is a version number in order not to erase the previous trace.

3.3.11 Reexecuting a run

`INTERPRET`

Interprets a trace file. The prefix is given by the user.

3.3.12 Moving to the CONSTRUCTION application

`CONSTRUCTION`

Moves to the CONSTRUCTION application. After the move, the selection mode is MOUSE P.

3.3.13 Moving to the PREP_MESH application

`PREP_MESH`

Moves to the PREP_MESH application. After the move, the selection mode is MOUSE P.

3.3.14 Moving to the EDIT_MESH application

`EDIT_MESH`

Moves to the EDIT_MESH application. When the move occurs and after the 4 questions have been answered, the program triangulates all the defined domains. If no domain has been defined, everything is meshed. The triangulation depends on 4 parameters. To get the default value, type `(CR)`for each parameter.

1. a *debug* parameter (default=0) that defines two other parameters:

    - impression =debug mod 10 : 0 => no printing, 9 => a lot of printing

- graphic debugging = debug / 10 : 0 => no debugging, 9 => all the phases of the meshing algorithm are displayed (*this is very slow*).

2. a *puis* parameter of refinement depth. The smaller *puis* is, the larger will refinement zones develop. $puis = .25$ is the default value. Reasonable values are between .1 and 10.

3. a *coef* parameter of triangle density. The number of triangles generated is in $O(coef^2)$. $coef = .75$ is the default value.

4. furthermore, the system asks whether a regularization is to be applied (default is yes);

After the move, the selection mode is `MOUSE P`.

3.4 The screen management menu

The screen management menu is shown in figure 3.6. It is present in all applications. It is used to handle the display of both databases. The display context consists in storing a display mask. This mask is a rectangle in subject space, computed so as to preserve the same scales in X and Y. The display context stack is in fact an 8 level circular stack. It is initialized as: scale 1, mask centered at the origin.

ZOOM +
ZOOM -
TRANSLATION
SCALE
REFRESH
C_MASK
PREVIOUS
NEXT
SHOW_ALL

Figure 3.6: The screen management menu

The operations of the screen management menu are:

3.4.1 Enlarging the display

`ZOOM + <COORD> <COORD>`

The rectangle defined by the two `<COORD>` is displayed full screen. The new display context is stacked in the display stack. In general, we click two times to define two opposite corners of the rectangle.

3.4.2 Reducing the display

`ZOOM - <COORD> <COORD>`

The contents of the full screen is displayed in the rectangle defined by the two `<COORD>`. The new display context is stacked in the display stack.

3.4.3 Translating the graphic window

`TRANSLATION <COORD> <COORD>`

Translates the graphic window by vector going from the first `<COORD>` to the second `<COORD>`. The new display context is stacked in the display stack.

3.4.4 Defining a scale

`SCALE`

Changes the display scale without moving the center of the window. The new display context is stacked in the display stack.

3.4.5 Redrawing the graphic window

`REFRESH`

Redraws the current display. This operation is very useful to erase unwanted residues that are often present.

3.4.6 Centering the display

`C_MASK <COORD>`

Centers the window on the selected point `<COORD>`. The new display context is stacked in the display stack.

3.4.7 Redrawing the previous display

`PREVIOUS`

Goes down one level in the display context stack. The previous display context is restored.
Note: this context always exists because all eight levels are initialized by default.

3.4.8 Redrawing the next display

`NEXT`

Goes up one level in the display context stack. The next display context is used, see note above.

3.4.9 Drawing the whole display

`SHOW_ALL`

Computes the mask so as to show the whole database and displays it full screen.

Chapter 4

The construction application

4.1 Presentation

This application is controlled by the application menu on top of the screen (figure 4.1) and the selection menu on bottom of the screen (figure 3.1). It is used to build and manipulate 2D objects made up of points, segments, arcs and splines. Circles and straight lines are only used as construction aids.

- The application uses elementary geometry to define points, straight lines, circles, segments and arcs. For example, by using the tangency property, it is possible to define a circle tangent to three straight lines or a straight line tangent to two circles.
- The application uses splines—a series of points—to define open or closed curves.
- The application is able to cut one object with another object, except the circles and straight lines, which are only construction aids. It is for example possible to cut an object by a straight line. However, the straight line will not be affected by this operation.
- The application is able to round off angles.
- The application is able to duplicate an object n times by using the classical affine transformations: symmetry, rotation, homothety.
- The application is able to *"outline"* figures.
- The application is able to invert arcs and segments.
- etc.

POINT	CIRCLE	ARC	ROUND	CENTER	DISTANCE	SWAP	→ → →
STRT LINE	SEGMENT	SPLINE	OUTLINE	RADIUS	ANGLE	INVERT	→ → →

← ← ←	COMPLE_ARC	CUT	ADD	RATIO	ROTATION	SYMMETRY
← ← ←	MERGE	CHANGE	IDEM	NUMBER	TRANSLAT	HOMOTHETY

Figure 4.1: The CONSTRUCTION application menu

4.2 Utilization

Let us recall that points, straight lines, circles, arcs, segments, splines, coordinates (current points, endpoints, etc.) can be selected via the selection menu. We will thus denote by `<POINT>`, `<LINE>`, `<CIRCLE>`, `<SEGMENT>`, `<ARC>`, `<SPLINE>`, `<COORD>` the operation of selecting one of these elements or part of it. Furthermore, a `<VALUE>` is created by using the calculator or its shortcuts, see sections 3.1,3.2 for more detail.

4.2.1 Meta-syntax of description of the operations

In the meta-syntax of description of the construction language, we will note `<xxx>` a series of operations defined by `<xxx> = ... . ;`

We will use the following symbols:

- `A  B` to denote the concatenation of `A` and `B` (and) "*we wait for* `A` *then* `B`".
- `A|B` to denote an alternative (or) "*we wait for* `A` *or* `B`" (this operation has less priority than concatenation).
- `*` to repeat the ensuing expression 0 or several times.
- `"` to denote nothing, which in all cases is used to achieve something.
- parentheses `( )` to change priorities.

Example: The expression: `<xxx> = ( A | ") *(B|C) E ;` can be used to enter character strings beginning or not by `A`, composed of a series of `B` or `C` and ending with `E`.

Examples of valid strings:

```
ABBCCCCE          E    AE    CBCBCBCBCBCBCBE
ABBBBBBBE         BE   ABE   BCBCBCBCBCBCBCE
BBBBBBBBBE        CE   ACE   CCCCCCCCCCCCCCE
```

We will call

- `<point_coor>` the selection of `<POINT>` or `<COORD>`;
- `<constraint>` the selection of `<POINT>`, `<COORD>`, `<LINE>`, `<CIRCLE>`, `<SEGMENT>` or `<ARC>`;
- `<elements>` the selection of `<POINT>`, `<COORD>`, `<LINE>`, `<CIRCLE>`, `<SEGMENT>`, `<ARC>` or `<SPLINE>`.
- etc.

In the meta-syntax, this reads:

```
<point_coor> = <POINT>      | <COORD>;
<constraint> = <point_coor> | <LINE> | <CIRCLE> | <ARC>
                            | &lt;SEGMENT&gt;;
<element>    = <constraint> | <SPLINE>;
```

In the definition of `<constraint>`, the ARC is identified with its supporting circle. Similarly, the SEGMENT is identified with its supporting straight line.

Remark: In the database the elements are described by:

POINT	(a,b) with: x=a; y=b
LINE	(a,b,c) such that ax+by+c=0
CIRCLE	center(a,b) and radius r
SEGMENT	(P1,P2) with P1=(x1,y1) and P2=(x2,y2)
ARC	center at P1 passing through P2 and with angle α, with P1=(x1,y1) and P2=(x2,y2)
SPLINE	(number of points, head of the list of points)

At the level of the CONSTRUCTION application, all menus other than the CONSTRUCTION menu are filtered through the applications SELECTION, CALCULATOR, GENERAL and GRAPHIC SCREEN MANAGEMENT. Thus clicking in one of these menus has no direct effect on the application, except possibly changing the selection mode or entering a value. Only the boxes of the application menu and the selections govern the CONSTRUCTION application. For this reason, we will denote by `XXXX` the operation of clicking in the XXXX box of the current application menu. There will be no confusion with boxes of the same name in other menus.

Example:

- the operation of selecting a circle is denoted `<CIRCLE>`
- the operation of clicking in the CIRCLE item of the CONSTRUCTION menu is denoted `CIRCLE`

Those two "circles" must not be confused with each other. Similarly, if we click in the CENTER item of the selection menu, we will select the center of an arc or a circle which produces a `<COOR>`. On the other hand, if we click in the CENTER item of the CONSTRUCTION menu (top), we obtain `CENTER`.

4.2.2 Modification of the state of the system

The state of the system is a set of variables used by the application. The global variables of the application are:

- %DISTANCE used to define a signed distance which is useful for constructing points, lines, segment, etc.
- %RADIUS used to define the radius of an arc or circle, etc.
- %ANGLE used to define the angle of an arc or between 2 lines, etc.
- %RATIO used to define the ratio of an homothety, etc.
- %NUMBER used to define the number of duplications during transformations.

These variables can be modified at almost any moment.

We denote by `<state modif>` a modification of the state of the system, which is to say a modification of the value assigned to one of the variable.

```
<state modif>  = *( <distance> | <radius> | <angle> | <ratio> | <number> )
```

- Modification of %DISTANCE: `<distance>= DISTANCE (<VALUE> | IDEM)`
- Modification of %RADIUS: `<radius> = RADIUS (<VALUE> | IDEM)`
- Modification of %ANGLE: `<angle> = ANGLE (<VALUE> | IDEM)`
- Modification of %RATIO: `<ratio> = RATIO (<VALUE> | IDEM)`
- Modification of %NUMBER: `<number> = NUMBER (<VALUE> | IDEM)`

IDEM saves the previous value of the state variable.

4.2.3 Construction of points

```
POINT
      *(  <COORD>
        | <POINT>  (  <CIRCLE> | <LINE>
                    | <ARC>    | &lt;SEGMENT&gt; | <SPLINE> )
        | <LINE>   <element>
        | <CIRCLE>   <element>
        | <SPLINE>   <element>
       );
```

- First alternative, `<COORD>`: Creation of the point of current coordinates by operating the selection menus: `MOUSE P`, `XY POINT`, `XY FILE`, `INTERS`, `EXTREM`, `MIDDLE`, `CENTER`.
- Other alternatives: We construct a point after having selected two elements.

- If one of the elements is a `<POINT>` let point P be the projection of this `<POINT>` on the other element
  * if this other element is a `<line>` we create a point at distance %DISTANCE from P on the `<line>` in the direction of the `<line>`
  * otherwise we create point P.
- If none of the two elements is a `<POINT>`, we create the intersection point of the two elements. Since there are in general two points in the intersection of a `<line>` and a `<circle>`, or of two `<circle>`, we choose "our" point by retaining the one closest to the points used to select the elements.

Examples of point construction

We start by clicking in the POINT box of the construction menu to switch to point construction mode. Then

- we click in the `MOUSE P` box of the selection menu. After that, each click in the graphic window constructs a point.
- we click in the `XY POINT` box of the selection menu. Each time we enter a couple of numbers on the calculator—*typed on the keyboard and followed by = or* `(CR)`*—we construct a point.*
- *we click in the* `XY FILE` *box and we enter a file name. A point is constructed for each couple of numbers in the formatted file.*
- *we construct points of intersection between two elements (line, segment, circle, arc, spline) by selecting a couple of such elements.*

4.2.4 Construction of straight lines

```
LINE  *(  <constraint> <constraint>
           | IDEM *&lt;SEGMENT&gt;
         );
```

If the constraints are (the order is irrelevant):

- `<CircleOrPoint>` := `<circle> | <point_coor>`;
- `<CircleOrPoint> <CircleOrPoint>`: the line created is tangent to the two `<circle>` at the points closest to the ones used to select the two `<circle>`
- `<line> <line>`:
  - if the two lines intersect, the line created is the bisecting line of the two others.
  - if the two lines are equal, we create a line that is parallel to them and at distance %DISTANCE on the side of the selection point.
  - otherwise we create a line parallel to the two `<line>` at the same distance from both.
- `<line> <circle>`: the line created is parallel to the `<line>` and tangent to the `<circle>` at a point closest to the one used to select the `<circle>`.
- `<line> <point_coor>`: the line created makes an angle %ANGLE with the `<line>` and passes through the `<point_coor>`.

In the other alternative (IDEM), we construct the line that supports the *`<segment>`.

4.2.5 Construction of circles

```
CIRCLE
  *(  <radius> *( CENTER <point_coor> | <constraint> <constraint>)
    | <constraint> (  <radius>  <constraint>
                   | <constraint> ( <radius> | <constraint> ))
    | CENTER <point_coor> *(<radius> | <constraint>)
    | IDEM *<ARC>
   );
```

Explanation of the various cases (the order is irrelevant):

- `<radius> CENTER <point_coor>`: circle of center at point `<point_coor>` and radius %RADIUS.
- `<radius> <constraint> <constraint>`: circle of radius %RADIUS and constrained to be tangent to the two selected elements.
- `<constraint> <constraint> <constraint>`: circle constrained to be tangent to the three selected elements.
- `CENTER <point_coor> <constraint>`: circle with a given center and tangent to the selected constraint.
- `IDEM` $*$ `<arc>`: circle supported by the $*$ `<arc>` (same center and same radius as the $*$`<arc>`).

4.2.6 Construction of arcs

```
ARC *(  CENTER <point_coor> *(   <angle> <constraint>
                            | <radius>
                            | <constraint> (<angle> | <constraint>))
     | <radius> *(  CENTER <point_coor>
                 | <constraint> <constraint> )
     | <constraint> (   <radius> <constraint>
                     | <constraint> (<radius> | <constraint>))
     | IDEM *<CIRCLE>
    );
```

Explanation of the various cases (the order is irrelevant). To construct an ARC, we first create a circle satisfying the same `<constraint>`s and then we create an arc on this circle.

- `CENTER <point_coor> <angle> <constraint>`: arc with a given center starting at the point where it is tangent to the `<constraint>` and with an angle `<angle>` from this point.
- `CENTER <point_coor> <radius>`: arc with a given center, given radius %RADIUS and angle 2π
- `CENTER <point_coor> <constraint> <constraint>`: arc with a given center starting at the point where it is tangent to the first `<constraint>`, ending at the level of the point where it is tangent to the second `<constraint>` and turning counterclockwise.
- `<radius> <constraint> <constraint>`: arc with a given radius %RADIUS starting at the point where it is tangent to the first `<constraint>`, ending at the point where it is tangent to the second `<constraint>`.
- `<constraint> <constraint> <constraint>`: arc tangent to the three `<constraint>` and linking the three points of tangency in the order of the `<constraint>`.
- `IDEM` $*$`<circle>`: creates an arc of angle $2\ \pi$ on each selected `<circle>`.

4.2.7 Construction of segments

```
SEGMENT *( <constraint> <constraint> );
```

Explanation of the various cases (the order is irrelevant).

To construct a SEGMENT, we first create the LINE that has the same constraints. Then we construct a segment on this line connecting the two points of tangency to the two constraints if they exist. Otherwise, we use the state variable %DISTANCE.

Thus for example:

- `<circle> <circle>`: The segment is tangent to the two `<circle>` (at the points closest to the points used to select the two `<circle>`).
- `<line> <line>`:
  - if the lines intersect, the segment is the bisecting line of the two `<line>` and is of length %DISTANCE.
  - if the two lines are equal, we create a segment that is parallel to them and at distance %DISTANCE on the side of the selection point. The segment is of length %DISTANCE or of the same length as `<line>` depending on whether `<line>` is a LINE or a SEGMENT.
  - otherwise we create a segment parallel to the two `<line>` at the same distance from both. It is of length %DISTANCE.
- `<line> <circle>`: The segment is parallel to the selected `<line>` and is tangent to the `<circle>` at the point closest to the one used to select the `<circle>`. It is of length %DISTANCE and starts at the point of tangency with the `<circle>`.

4.2.8 Construction of splines

```
SPLINE <point> *<point>;
```

Constructs the cubic spline passing through the `<points>`. We can create a closed spline by making the last point equal to the first point (by selecting it).

4.2.9 Geometrical transformations

ROTATION, TRANSLAT, HOMOTHETY, SYMMETRY all define transformations that can be applied on selected elements.

```
ROTATION    <state modif> ( CENTER | " ) <point_coor> <state modif> *<element>
TRANSLAT    <point_coor> <point_coor>    *<element>
HOMOTHETY  <state modif> CENTER (<point_coor> | " ) <state modif>   *<element>
SYMMETRY   (<point_coor> | <LINE>| &lt;SEGMENT&gt;)   *<element>
```

ROTATION applies the rotation of center `<point>` and angle %ANGLE to the `<elements>`. If %NUMBER ≤ 1, the `<elements>` are moved, otherwise %NUMBER-1 copies of each `<element>` are created. Therefore, after transformation there are %NUMBER `<elements>` on the screen.

TRANSLAT applies to the `<elements>` the translation by a vector defined by the first `<point>` and the second `<point>`. If %NUMBER ≤ 1, the `<elements>` are moved, otherwise %NUMBER-1 copies of each `<element>` are created. Therefore, after transformation there are %NUMBER `<elements>` on the screen.

HOMOTHETY applies the homothety of center `<point>` and ratio %RATIO to the `<elements>`. If %NUMBER ≤ 1, the `<elements>` are moved, otherwise %NUMBER-1 copies of each `<element>` are created. Therefore, after transformation there are %NUMBER `<elements>` on the screen.

SYMMETRY applies the symmetry with respect to `<point>` or `<line>` on all selected `<elements>`. Warning: if %NUMBER is larger than 2, elements will be superimposed on one another.

4.2.10 Rounding off angles

This command is used to round off angles between SEGMENTS and ARCS.

```
ROUND  *( <constraint> <constraint> )
```

We round off an angle between the two `<constraint>` with radius %RADIUS, *the rounded angle will always turn counterclockwise*. A rounded angle is an arc whose supporting circle is tangent to the two `<constraint>`. The arc starts at the first point of tangency and ends at the second point of tangency, turning counterclockwise. If one of the `<constraint>` is an ARC or a SEGMENT, this `<constraint>` is modified in such a way that the element terminates at the point of tangency with the rounded angle.

4.2.11 Defining a contour

```
CONTOUR   (<LINE> | <CIRCLE>)  (<LINE> | <CIRCLE>)
             (<LINE> | <CIRCLE>) (COMPLE_ARC | '')
          *( (<LINE> | <CIRCLE>) (COMPLE_ARC | '')      )
```

This command is used to define a contour formed by SEGMENTS and ARCS supported by a set of straight lines and circles. The program functions by calculating the intersection of constraint i with constraint $i+1$, which gives point I, then the intersection of constraint $i+1$ with constraint $i+2$, which gives point I+1. I and I+1 form a new portion of the contour and so on.

Let us give an example with straight lines (figure 4.2):

Let us assume that we have five lines D_1 D_2 D_3 D_4 D_5 that intersect at p_1 p_2 p_3 p_4 p_5, and that we wish to define the contour composed of the segments (p_1, p_2), (p_2, p_3), (p_3, p_4), (p_4, p_5) and (p_5, p_1). We denote by +i the ith selected point. We select D_2 at +2 then D_1 at +1 which defines p_1

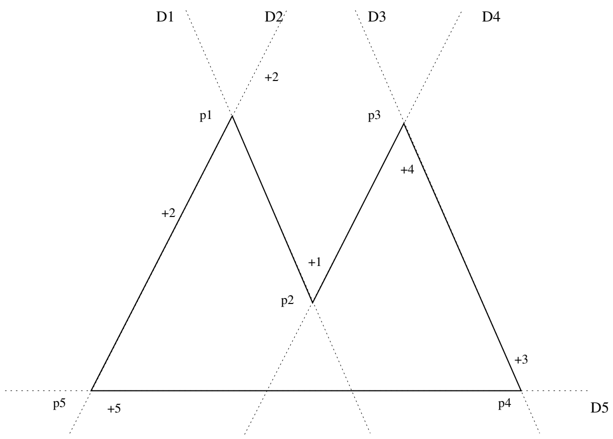

Figure 4.2: Example of contour definition

Then we select D_4 at +4 which defines p_2 (at this point, the program draws (p_1, p_2)
Then we select D_3 at +3 which defines p_3 (at this point, the program draws (p_2, p_3)
Then we select D_5 at +5 which defines p_4 (at this point, the program draws (p_3, p_4)
Then we select D_2 at +2 which defines p_5 (at this point, the program draws (p_4, p_5)
Then we select D_1 at +1 which defines p_1 (at this point, the program draws (p_5, p_1)

In the case of circles, we must be careful because the selection points are used to remove ambiguities of multiple intersections: the selection point of the i constraint is used to remove the selection ambiguity between constraints i and i+1.

4.2.12 Reversing segments or arcs

```
REVERSE *( <ARC> | &lt;SEGMENT&gt; )
```

Reverses a SEGMENT or an ARC

- ARC : $\alpha \rightarrow -\alpha$
- SEGMENT: $(P1, P2) \rightarrow (P1, 2 * P1 - P2)$ (Symmetry / P1)

4.2.13 Complementing arcs

```
COMPLE_ARC *ARC
```

Changes an ARC into its complement if $(\alpha < 0)$ then $\alpha = \alpha + 2\pi$ otherwise $\alpha = \alpha - 2\pi$

4.2.14 Inverting elements

```
INVERT *(&lt;ARC&gt; | &lt;SEGMENT&gt; | &lt;SPLINE&gt;)
```

Inverts an ARC, SEGMENT or a SPLINE

- ARC : $(C, P, \alpha) \rightarrow (C, Rot(P, \alpha)/C, -\alpha)$
- SEGMENT: $(P_1, P_2) \rightarrow (P_2, P_1)$
- SPLINE : $(P_1, P_2, ..., P_n) \rightarrow (P_n, P_{n-1}, ..., P_2, P_1)$

4.2.15 Cutting one element with another?

```
CUT  * (  &lt;point_coor&gt; *(&lt;ARC&gt; | &lt;SEGMENT&gt; | &lt;SPLINE&gt;)
          | (&lt;LINE&gt; | &lt;CIRCLE&gt; | &lt;ARC&gt; | &lt;SEGMENT&gt; | &lt;SPLINE&gt;)
            (&lt;LINE&gt; | &lt;CIRCLE&gt; | &lt;ARC&gt; | &lt;SEGMENT&gt; | &lt;SPLINE&gt;)
         )
```

If we first select `<point>`, we cut an ARC, a SEGMENT or a SPLINE in two by the projection of `<point>` on the element.
For the SPLINE, we cut at the definition point closest to point `<point>`.
Otherwise we cut the selected elements with one another if they are divisible (non divisible elements are LINE and CIRCLE).

4.2.16 Changing part of an element

```
CHANGE *(  ANGLE <ARC> (<VALUE> | IDEM)
         | RADIUS <CIRCLE> (<VALUE> | IDEM)
         | <POINT> <point_coor>)
```

This command is used to change :

- the angle of a selected ARC
- the radius of a CIRCLE
- the position of a `<point>` (the first point is changed into the second one. The first point can be the center of an arc or a circle)
- the extremity of a SEGMENT or an ARC.
- a definition point of a SPLINE, thus the shape of the SPLINE.

4.2.17 Adding elements

```
ADD *(  POINT   *( <ARC> | &lt;SEGMENT&gt; | <SPLINE> | <POINT> <POINT> )
      | SEGMENT *( &lt;SEGMENT&gt; &lt;SEGMENT&gt; )
      | SPLINE  *( <SPLINE> (   <SPLINE> | *<POINT>  ) )
      | ARC     *( <ARC> <ARC>  )
```

- If POINT, we add %NUMBER points with a ratio %RATIO on ARC or SEGMENT or SPLINE or between the two POINTS.
- If SEGMENT, we add %NUMBER segments defined by linear interpolation of ratio %RATIO between the two SEGMENTS.
- If SPLINE
  - followed by the selection of two SPLINES, then we add %NUMBER splines defined by linear interpolation of ratio %RATIO between the two SPLINES.
  - followed by the selection of one SPLINE and one POINT, then we add new definition POINTS given by *`<point>` to the selected SPLINE.
- If ARC, we add %NUMBER arcs defined by 3 points (2 extremities and the middle point) defined by linear interpolation of ratio %RATIO between the extremities and the middle points of the 2 selected arcs.

4.2.18 Merging two elements

```
MERGE *( &lt;segment&gt; &lt;segment&gt; | <arc> <arc> | <spline> <spline> )
```

This command is used to merge 2 SEGMENTS, 2 ARCS or 2 SPLINES that have a common extremity into a single element of the same type. The resulting element has the same orientation as the first selected element. This instruction is only executed if the following conditions are met:

- in the case of 2 segments, both segments must be colinear.
- in the case of 2 arcs, both arcs must have the same center and the same radius.

Chapter 5

The PREP_MESH application

5.1 Presentation

This application is controlled by the application menu on top of the screen (figure 5.1) and the selection menu on bottom of the screen (figure 3.2). It is used to define:

- the discretization of the contour lines,
- the subdomains to be meshed,
- the references of the lines and their extremities, with the objective of taking boundary conditions into account (cf. the finite elements method),
- the type of mesh generator, either structured *(deformed quare)* or non structured *(Voronoï)*, for each subdomain.

Furthermore, at this point, we have the option of interfacing with the APNOPO mesh generator by generating a data file for the APNOXX preprocessor, see [7].

LINE	COMPONENT	UNCRACK	RATIO	NODE_REF	DOMAIN_REF	→ → →
DOMAIN	ALL	CRACK	Nb_INTERVALS	LINE_REF	Lg_INTERVAL	→ → →

← ← ←	DOMAIN_DEF	QUADRANGLE	REGULAR	REMOVE	SEE	GENERATE
← ← ←	INTERIOR	TRIANGLE	CORNERS	VERIFY	STRIP	Gen. Base

Figure 5.1: The PREP_MESH application menu

5.2 Utilization

When we enter this application, the program automatically calculates all the interior and exterior connex components of the design, for example figure 5.2. The interior connex components are oriented counterclockwise and the exterior connex components are oriented clockwise.

We thus obtain automatically the following connex components:

```
P5,P3,P4       >0
```

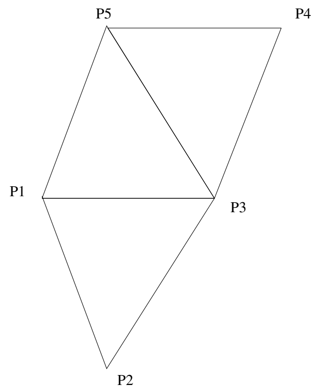

Figure 5.2: Example of components

```
P1,P3,P5       >0
P1,P2,P3       >0
P1,P5,P4,P3,P2 <0
```

- We will call <element> an ARC, a SEGMENT or a SPLINE.
- We will call <component> a component selected through one of its <element>.
- We will call <domain> a domain selected through one of its elements (<component> or interior <element>).
- We will call <line> = LINE <element1> <element2> the set of all elements going from the first element <element1> to the second element <element2> in the orientation of the component to which both elements belong.

5.2.1 Modification of the state of the system

The state of the system is a set of variables used by the application. The global variables of the application are:

- %RATIO, used to define the ratio of the discretization of a curve (line, segment, arc or spline)
- %NBINTR, used to define the number of intervals of discretization of a curve
- %LGINTR, used to define the length of the of intervals of discretization of a curve
- %NUREF, used to define a reference number for a point, a curve or a subdomain.

These variables can be modified at almost any moment.

We will call <state modif> a modification of the state of the system, which is to say a modification of the value assigned to one of the variables.

```
<state modif>  = *( <ratio> | <nb interval> | <lg interval> | <ref number> )
```

- Modification of %RATIO: <ratio> = RATIO (<VALUE> | IDEM)
- Modification of %NBINTR: <nb interval> = NBINTR (<VALUE> | IDEM)
- Modification of %LGINTR: <lg interval> = LGINTR (<VALUE> | IDEM)
- Modification of %NUREF: <ref number> = NUREF (<VALUE> | IDEM)

IDEM keeps the previous value of the state variable.

5.2.2 Definition of intermediate points

```
Nb_INTERVAL * <VALUE>
              * (   LINE       *(( <VALUE> | ")<element1> <element2>)
                  | COMPONENT  *(( <VALUE> | ")<element3>)
                  | ALL
                  |              ( <VALUE> | ")<element4>  )
```

This command is used to generate intermediate points on

- the elements of a `<line>` defined by `<element1> <element2>`
- the elements of a `<component>` selected by `<element3>`
- all the elements of the D.B.
- only one element `<element4>`

so as to obtain %NB_INTERVAL intervals. If `<VALUE>` is entered, %NB_INTERVAL takes this value and it is displayed on the bottom line of the screen.

```
Lg_INTERVAL  *<VALUE>
              * (    LINE        *(( <VALUE> | ")<element1> <element2>)
                  | COMPONENT    *(( <VALUE> | ")<element3>)
                  | ALL
                  |                ( <VALUE> | ")<element4>  )
```

This command is used to generate intermediate points on

- the elements of a `<line>` defined by `<element1> <element2>`
- the elements of a `<component>` selected by `<element3>`
- all the elements of the D.B.
- only one element `<element4>`

in such a way that the distance between two consecutive points is close to %Lg_INTERVAL. If `<VALUE>` is entered, %Lg_INTERVAL takes this value and it is displayed on the bottom line of the screen.

5.2.3 Definition of the ratio

```
RATIO  * <VALUE>
         * (   LINE      *((  <VALUE> | ")<element1> <element2>)
             | COMPONENT *((  <VALUE> | ")<element3>)
             | ALL
             |              ( <VALUE> | ")<element4>  )
```

This command is used to assign %RATIO as ratio to

- the elements of a `<line>` defined by `<element1> <element2>`
- the elements of a `<component>` selected by `<element3>`
- all the elements of the D.B.
- only one element `<element4>`.

If `<VALUE>` is entered, %RATIO takes this value and it is displayed on the bottom line of the screen.

5.2.4 Definition of the reference number of extremities

```
NODE_REF * <VALUE>
         * (   LINE      *(( <VALUE>  | '')<element1> <element2>)
             | COMPONENT *(( <VALUE>  | '')<element3>)
             | DOMAIN    *(( <VALUE>  | '')<element4>)
             | ALL
             |              ( <VALUE>  | '')<element5>  )
```

This command is used to assign %NUREF as reference number to

- the extremities of the elements of a `<line>` defined by `<element1>` `<element2>`
- the extremities of the elements of a `<component>` selected by `<element3>`
- the extremities of the elements of a `<domain>` selected by `<element4>`
- all extremities of the elements of the D.B.
- the extremities of only one element `<element5>`

If `<VALUE>` is entered, %NUREF takes this value and it is displayed on the bottom line of the screen.

If the selected elements are cracks, `<VALUE>` will be assigned to the extremity and on the side where the element, the LINE or the COMPONENT was clicked, and by continuity to the extremities of the adjacent elements. In effect, a crack element possesses a line reference number and two extremity reference numbers on its left and/or on its right. These numbers can be identical or different. If they are different, then the element is necessarily a crack.

5.2.5 Definition of the line reference number

```
LINE_REF * <VALUE>
         * (   LINE      *(( <VALUE>  | '')<element1> <element2>)
             | COMPONENT *(( <VALUE>  | '')<element3>)
             | DOMAIN    *(( <VALUE>  | '')<element4>)
             | ALL
             |              ( <VALUE>  | '')<element5>)
```

This command is used to assign %NUREF as reference number to

- the elements of a `<line>` defined by `<element1>` `<element2>`
- the elements of a `<component>` selected by `<element3>`
- the elements of a `<domain>` selected by `<element4>`
- all the elements of the D.B.
- only one element `<element5>`.

If `<VALUE>` is entered, %NUREF takes this value and it is displayed on the bottom line of the screen.
If the selected elements are cracks, %NUREF will be assigned to the side on which the element, the LINE or the COMPONENT was clicked.

5.2.6 Cracking lines

```
CRACK   * (   LINE      *(<element1><element2>)
            | COMPONENT * <element3>
            | ALL
            |              <element4>  )
```

This command is used to crack a line in a mesh (the two side of a line a different in the mesh)

- the elements of a LINE defined by `<element1>` `<element2>`

- the elements of a COMPONENT selected by `<element3>`
- all the elements of the D.B.
- only one element `<element4>`.

5.2.7 Uncracking lines

```
UNCRACK * (   LINE      *(<element1><element2>)
              | COMPONENT * <element3>
              | ALL
              |             <element4>  )
```

This command is used to uncrack

- the elements of a LINE defined by `<element1>` `<element2>`
- the elements of a COMPONENT selected by `<element3>`
- all the elements of the D.B.
- only one element `<element4>`.

This the inverse of the cracking operation. The number of references is brought down to 1 by only retaining the left reference number of the crack.

5.2.8 Definition of a subdomain reference number

```
DOMAIN_REF *( <VALUE>  | <domain> | ALL )
```

This command is used to assign `<VALUE>` as subdomain reference number to the selected domain `<DOMAIN>` or to all defined domains. If `<DOMAIN>` does not exist, it is created.

5.2.9 Defining a domain

```
DEF_DOMAI <element1>
          * (<element2> | INTERIOR *(<point> | <element3> | <line>) )
```

- If `<element1>` is not already included in a domain, then we create a domain with number %NUREF that has `<element1>` as exterior component and whose interior components are selected by the following elements `<element2>` if any or rather, the interior elements are defined by INTERIOR followed by `<point>` or the intermediate points of `<element3>` or the intermediate points of the elements of the `<line>`.
- If `<element1>` is already included in a domain then the component selected by `<element2>` is added to the components of the domain as an interior component.

5.2.10 Adding interior elements to a domain

```
INTERIOR *( <element1> *( <point> | <element2> | <line>) )
```

The INTERIOR command is used to add interior elements to a domain selected by the element `<element1>`. The interior elements that can be added to a domain are `<point>` or the intermediate points of `<element2>` or the intermediate points of the elements of `<line>`.

5.2.11 Verification of the correct definition of a domain

```
VERIFY  *( ALL | <element>)
```

- This command is used to verify if the domain selected by `<element>` has been defined correctly.
- It is used to verify if all domains are defined correctly, for example to check that no interior components of negative surface are present, etc.

5.2.12 Removing elements from a domain

```
REMOVE  *(   INTERIOR *( <point> | <element1>)
             | DOMAIN *<element2>
             | ALL
             | <element3> )
```

This is used to remove

- the element `<element1>` or an interior `<point>` in a domain selected by this `<element1>` or this `<point>`
- a domain selected by `<element2>`
- all domains
- the component selected by `<element3>` of a domain. It is not possible to apply this command if the component is exterior because the first hole component would then become exterior, cf. internal structure of the data.

5.2.13 Meshing mode of a domain

```
TRIANGLE    *<domain>
```

Requests that the `<domain>` be divided into triangles by using a Voronoï algorithm [7]

```
QUADRANGLE  *(<domain>
              (  <line>
               | <element>
               | (  EXTREMITY<element1> EXTREMITY<element2>
                    EXTREMITY<element3> EXTREMITY<element4> )
               | " ) )
```

Requires that the `<domain>` be homeomorphic to a square (deformed quadrangle). The command requests that the quadrangular subdomain should contain a structured, NxM finite difference type mesh like in figure 5.3. It will be divided into quadrangles, unless we change the partition mode with (5.2.14) if you want to use the GENERATE item 5.2.16. There are two ways of describing the quadrangular domain, by the four edges or the four vertices:

- the first edge of the domain is defined by line `<line>` or by element `<element>`,
- we enumerate the four vertices of the domain by giving the four extremities of the elements that define them. The vertices must be enumerated counterclockwise .

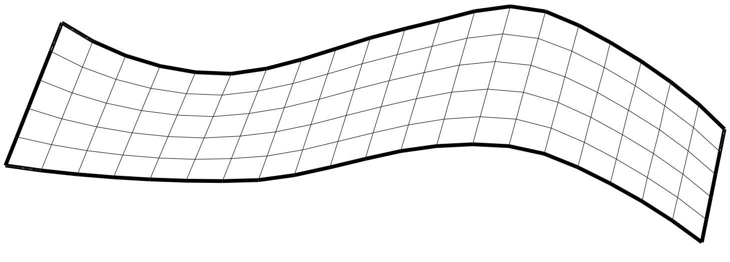

Figure 5.3: Example of QUANDRANGLE mesh: nb. points: on right and left edges is 6=N, and on lower and upper edge is 21=M

```
STRIP  *(<domain>
            (  <line>
             | <element>
             | (  EXTREMITY<element1> EXTREMITY<element2>
                  EXTREMITY<element3> EXTREMITY<element4> )
             | " ) )
```

Requires that the **<domain>** be homeomorphic to a square (deformed quadrangle) with an equal number of points N over a couple of opposite edges, like in figure 5.4 (the case **<quadrangle>** requires the equality of numbre of points over the two couple of opposites edges , the " quadragle mesh" generator is always used in this case). (N-2) internal lines are generated, joining correspondind points of opposite edges and points are distributed on these lines using interpolation procedures. These (N-2) internal lines definie (N-1) strips which are divided into triangle using a special option of the mesh generator (no extra internal points are generated).

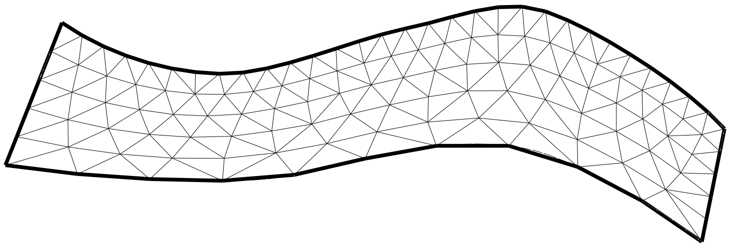

Figure 5.4: Example of STRIP mesh, nb. of node: on right and left edges is 6 = N , on lower edge is 11, and on upper edge is 31.

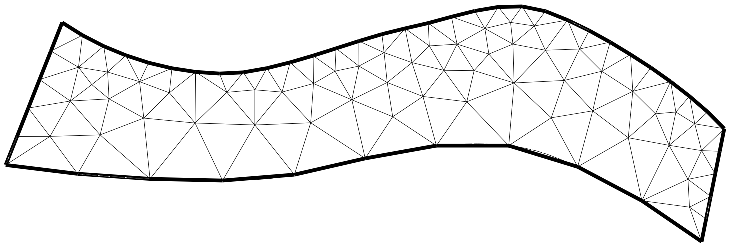

Figure 5.5: Example of Delaunay-Voronoïmesh with the same input than in the previous figure.

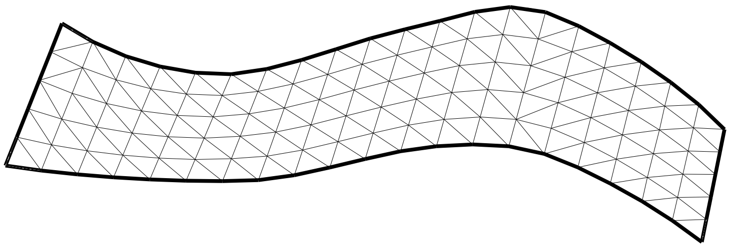

Figure 5.6: Example of STRIP mesh (Quadrangle mesh), nb. of nodes: on right and left edges is 6 = N , and on lower and upper edge is 21.

There are two ways of describing the strip domain, by the four edges or the four vertices:

- the first edge of the domain is defined by line `<line>` or by element `<element>`,
- we enumerate the four vertices of the domain by giving the four extremities of the elements that define them. The vertices must be enumerated counterclockwise .

Remark: This kind of mesh generator is may be useful of the boundary layer region.

5.2.14 Partition mode for a quadrangular domain

```
REGULAR  *(<domain> <VALUE> )
```

Requests that the partition of quadrangles into triangles in the `<domain>` be, see the MODULEF APNOPO booklet:

- regular on the right (value=2)
- regular on the left (value=1)
- non regular (value=0)

depending on the value.

```
CORNERS  *(<domain> <VALUE> )
```

Requests that the partition of quadrangles into triangles in the `<domain>` be effected taking or not the corners into account, depending on the value (Yes = 0 / No $\neq$ 0).

5.2.15 Visualization of domains or components

```
SEE    DOMAIN *(<domain> | <component> | ALL )
```

This command is used to visualize using thick lines the `<domain>` or the `<componant>` selected by one of its elements. This operation is useful for debugging a mesh (problems with intersection, round off error).

5.2.16 Generating an APNOXX input file

```
GENERATE
```

This command is used to generate an input file for APNOXX (Modulef).

- The input file will be assigned the name XXX by the user followed by the suffix .DATA.
- The input file will request APNOXX to generate a NOPO titled XXX.NOPO.

The program takes into account only those domains which have been previously defined.

Chapter 6

The EDIT_MESH application

6.1 Presentation

This application is controlled by the application menu (figure 6.1) on top of the screen and the selection menu (figure 3.3) at the bottom of the screen. It is used to edit a 2D mesh constructed from triangles and quadrangles, obtained either by switching from the **CONSTRUCTION** or **PREP_MESH** applications to the **EDIT_MESH** application, or by restoring a mesh (cf. general menus). Briefly, the possibilities are: adding, suppressing, moving the vertices of the mesh, reversing the edge that divides a quadrilateral, *quadrangulating, triangulating*, regularizing (placing the vertices at the barycenter of their neighbors), making the mesh Delaunay, modifying the references of a mesh (boundaries and subdomains), transforming a subdomain geometrically, renumbering, cracking.

REGULARIZE	DELAUNAY	ADD	MOVE	SUPPRESS	REVERSE	→ → →
RENUMBER	ALL	ORIGINALS	EVALUATE	ROTATION	TRANSLATION	→ → →

← ← ←	MODIF_REF	CRACK	TRIANGULATE	MIN ANGLE	MARK ELEMENT
← ← ←	SYMMETRY	HOMOTHETY	QUADRANGULATE	MAX ANGLE	BOUNDARY

Figure 6.1: The EDIT_MESH application menu

6.2 General remarks

The new types of selection of the application are:

- if the selection mode is ELEMENT, we select an element of type `<ELEMENT>` which is a "finite element" (triangle or quadrangle) (just click in the finite element);
- if the selection mode is EDGE, we select an element of type `<EDGE>` which is an edge of a mesh element (just click in the element that contains this edge or close to the latter);
- if the selection mode is VERTEX, we select an element of type `<VERTEX>` which is a vertex of a mesh element (just click in the element that contains this vertex or close to the latter);

- if the selection mode is S_DOM, we select an element of type `<S_DOM>` which is the connex component bounded by the interior and exterior boundaries and contains the clicked finite element.

Furthermore, we will often use the notation `<point>` to denote a point `<POINT>`, `<COORD>` or `<VERTEX>`.

```
<point> = <POINT> | <COORD> | <VERTEX>.
```

6.2.1 Definition of the state variables of the application

- `MARK_ELEMENT` is used to visualize the elements whose angles are out of bounds (minimum and maximum angles for triangles and quadrangles).
- `(TRIANGULATE|QUADRANGULATE) MIN_ANGLE *<VALUE>` is used to change the minimal value in degrees of angles for triangles or quadrangles.
- `(TRIANGULATE|QUADRANGULATE) MAX_ANGLE *<VALUE>` is used to change the maximum value in degrees of angles for triangles or quadrangles.
- `BOUNDARY` is used to move or not move the boundary points that are not extremities (flip flop). Initially, points cannot be moved with the MOVE or REGULARIZE operations.

6.3 Editing the mesh

The mesh editing operations are:

6.3.1 Moving a vertex

```
MOVE   *(<VERTEX>  <point>  |  BOUNDARY)
```

This command is used to move the selected vertex (which can be on a boundary depending on the value of the state variable `BOUNDARY`) to the position of the selected point, which can either be a *mouse_point* (defined with the mouse), an *xy_point* (defined with two calculator values (x,y)), a DB *point* or a *vertex* (in the latter case, the new vertex is located at the cursor position and not on the selected vertex).
Remark: if we move a boundary point, the point moves on the boundary.

6.3.2 Adding a vertex to the mesh

```
ADD *(<VERTEX>  |  <ELEMENT>)
```

This command is used to add a vertex in the mesh element that contains the selected point. If the selected element is a quadrangle, it is first split into two triangles. We add a vertex to the selected triangle by dividing it in three triangles. The coordinates of the new point are those of the cursor. Then the *triangular* mesh is made almost Delaunay.

6.3.3 Suppressing a vertex to the mesh

```
SUPPRESS  *(<VERTEX>  |  <S_DOM>)
```

This command is used to suppress:

1. either a non boundary point of the mesh and to remesh the hole thus created,
2. or a subdomain.

6.3.4 Reversing an edge of the mesh

```
REVERSE *( <EDGE>)
```

This command is used to reverse the internal edge of the quadrilateral defined by two adjacent triangles, on the condition that the triangles thus created be correct.

6.3.5 Delaunay mesh

```
DELAUNAY  *(<ELEMENT> |  <S_DOM>  |  ALL )
```

This command is used to force the mesh to satisfy the Delaunay criterion for triangles (the only vertices of the mesh contained in the circonscribed circle of a triangle are the vertices of this triangle),

1. either in the vicinity of the selected triangle,
2. or in the whole selected subdomain,
3. or in the whole mesh.

6.3.6 Regularizing the mesh

```
REGULARIZE  *(    <VERTEX>  |  <ELEMENT>  |  <S_DOM>
                |   ALL |  ORIGINALS |  BOUNDARY)
```

This command is used to regularize the mesh or a part of it. This means that we move all vertices of the part in question in turn to the barycenter of its neighbors. Boundary points are moved if this is permitted (cf. state variable).
Remark: if we move a boundary point, the point moves on the boundary.

6.3.7 Triangulating

```
TRIANGULATE *(  <ELEMENT>  |<S_DOM>  |   ALL |  ORIGINALS
             | MIN_ANGLE  | MAX_ANGLE  )
```

This command is used to triangulate the mesh or a part of it and to modify the minimum and maximum angle state variables of the triangles.

6.3.8 Quadrangulating

```
QUADRANGULATE    *(  <ELEMENT>   | <EDGE>  | <S_DOM>
                   |   ALL     |  ORIGINALS
                   | MIN_ANGLE  | MAX_ANGLE )
```

This command is used to *"quadrangulate"* the mesh or a part of it.

1. if an *element* is selected and if this elementis a triangle whose 3 adjacent elements are also triangles, then 3 quadrangles are created if these quadrangles are *valid*, i.e., convex and whose angles are between the minimum and maximum angles (cf. state variable).
2. if an *edge* is selected and if the 2 adjacent elements are triangles and if the edge is not the boundary of a subdomain, then a quadrangle formed from these 2 triangles is created if the quadrangle is *valid*.
3. if a *subdomain* is selected then the program will attempt to quadrangulate the entire subdomain (the method depends on the element selected to define the subdomain). *Warning: the entire subdomain and even already existing quadrangles will be re-quadrangulated.*
4. clicking the ALL item or the ORIGINALS item will cause the program to attempt *quadrangulating* the entire mesh (triangles might remain if the number of boundary edges is odd or if the angles are incorrect).
5. clicking the MIN_ANGLE item (resp. MAX_ANGLE) modifies the minimum angle state variable of the quadrangles (resp. maximum).

6.3.9 Cracking

```
CRACK *( <ARC> | &lt;SEGMENT&gt; | <SPLINE> )
```

This command is used to create or suppress cracks in the mesh. If the selected element of the DB (arc, segment or spline) is cracked, then it will be "uncracked", otherwise it will be cracked. A crack in a mesh is a line or a group of lines such that the nodes that are on one side or another of the line are different from each other. *In the graphic representations the cracked nodes are separated from one another to demonstrate the crack. However, this is a purely graphic device.*

6.3.10 Modifying the mesh references

```
MODIF_REF *( <VALUE>
             *(  <ARC>    | &lt;SEGMENT&gt;  |  <SPLINE>
               | <VERTEX> | <EDGE>     |  <S_DOM>)  )
```

This command is used to modify the references of different parts of the mesh.

Warning: the reference of a vertex is the reference of its supporting element in the DB (a point if it is an extremity) and similarly for edges. If the element is cracked, then there are two references, one on the left and the other one on the right.

6.3.11 Renumbering vertices

```
RENUMBER
```

This command is used to renumber the vertices of the mesh in order to decrease the bandwidth or profile of the corresponding Lagrange P_1 or Q_1 finite element matrix.

6.4 The transformations

The transformations apply either to subdomains or to the whole domain or only to the original subdomains, which is to say those that are drawn in continuous lines.

Note: a transformed, non original, subdomain is only defined by a pointer on the original subdomain and by the affine transformation (a 2×3 matrix). This implies that all modifications, moving, adding, suppressing, etc., of the original subdomain will affect the transformed subdomains. Moreover, the original subdomains are displayed in continuous lines and the tranformed subdomains are displayed in dotted lines. To effectively generate all the transformed subdomains, we must click in the `EVALUATE` box. The transformed subdomains then become original subdomains since we have evaluated the expression that defines the mesh.

Warning: as long as the tranformed subdomains are not evaluated, the DB boundary elements for these subdomains do not exist yet.

A point `<point>` will either denote a vertex or a mouse point, an xy point or a current point. The syntax is:

```
<a_transformation> ( ALL |  ORIGINALS | <S_DOM>*)
```

This means that after it is defined, a transformation will either apply to all subdomains or to the original subdomains (in continuous lines) or to explicitly selected subdomains.

A transformation is either a translation, a symmetry, a rotation or a homothety. Syntactically:

```
<a_transformation> =   <translation>  |  <symmetry>
                      | <rotation>    |  <homothety>;
```

which are defined as follows:

6.4.1 Translation

```
<translation> = TRANSLATION   <point>  <point>
```

Defines the translation going from the first point to the second point.

6.4.2 Symmetry

```
<symmetry> = SYMMETRY  (   <point_coor>  <point_coor>
                          | &lt;SEGMENT&gt;  | &lt;EDGE&gt;)
```

Defines the symmetry with respect to the straight line that supports the selected segment or edge, or that passes through the two selected points.

6.4.3 Rotation

```
<rotation> = ROTATION (<point>  <VALUE>  |  <VALUE>  <point>)
```

Defines the rotation around the point and with angle equal to the value entered on the calulator.

6.4.4 Homothety

```
<homothety>  = HOMOTHETY  (<point> <VALUE> | <VALUE> <point>)
```

Defines the homothety of center the point and ratio equal to the value.

6.4.5 Effectively generating the transformations

```
EVALUATE
```

This command is used to effectively generate all the transformed subdomains in order to edit them independently of their original. It also creates and transforms all the elements (segments, arcs, splines) that are referenced by the transformed subdomains.

6.4.6 An example of transformation

To create symmetrical subdomains we click in the SYMMETRY item, then we select either two points, a segment or an edge, then we select either all the subdomains to be transformed, or we click in the ALL item or in the ORIGINALS item. Finally, before saving the mesh under another format than "mesh", we click in the EVALUATE item.

Chapter 7

Example

7.1 The unit square with a hole

In this section, we show how to mesh a unit square with a hole. First of all, we run emc2, enter the number of the F3D graphic device and answer the few supplementary questions that are posed by the system *(this depends on the particular F3D implementation).*

7.1.1 Construction of the unit square with a hole

1. Click the `POINT` item of the construction menu
2. Click the `XY POINT` item of the selection menu
3. Enter `0=0=1=0=0=1=1=1=.5=.5=` on the keyboard *5 points are displayed in the middle of the screen since the scale is equal to 1*
4. Click the `SEE_ALL` item to see the five points clearly. The 5th point will be the center of the hole.
5. Click the `SEGMENT` item of the construction menu (on top of the screen. The bottom `SEGMENT` item belongs to the selection menu)
6. Click the `POINT` item of the selection menu (bottom)
7. Click in the graphic window on the extreme points of the 4 segments to be created. After each couple of clicks a segment is created. Therefore we must click 8 times, for example close to the following points in this order (0,0), (1,0), (1,0), (1,1), (1,1), (0,1), (0,1), (0,0)
8. Click the `CIRCLE` item of the construction menu (top) then click the CENTER item
9. Click in the graphic window close to the point (.5,.5) (we are still in point selection mode)
10. Click the `RADIUS` item
11. Enter `1/4=` on the keyboard: a circle of center (.5,.5) and radius 1/4 is displayed. *Warning: the circle as well as the straight line is only a construction aid. It must be transformed into an arc*
12. Click the `ARC` item of the menu to construct arcs
13. Click the `IDEM` item
14. Click the `CIRCLE` item of the selection menu to select the circle
15. Click in the graphic window. Since there is only one circle, it is selected. The circle being now useless, we destruct it.
16. Click the `DESTRUCT` item of the general menu (left)
17. Click in the graphic window. The circle and the arc disappear since they are superimposed . *Warning: we are still in destruction mode.*

18. Click the **REFRESH** item to display the arc

The construction of the geometry is done (figure 7.1).
We can switch to the PREP_MESH application.

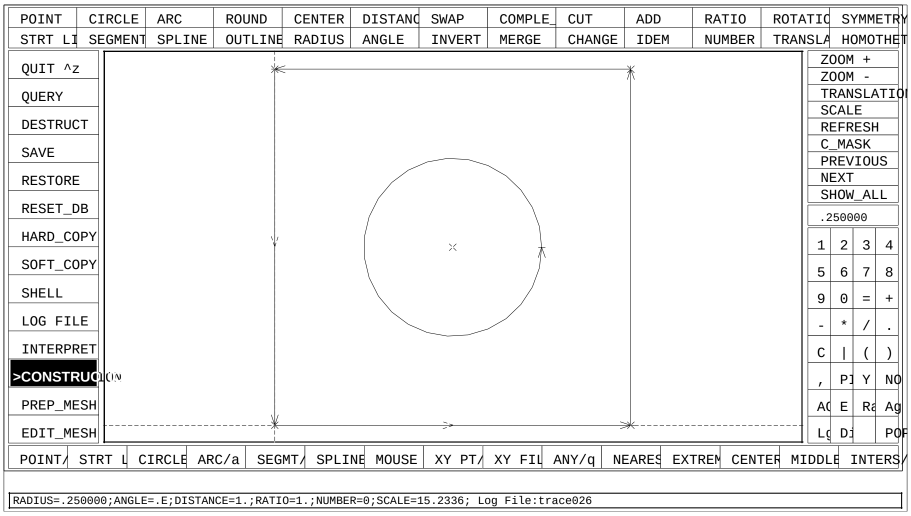

Figure 7.1: The geometry of the square domain with a hole

7.1.2 Definition of the boundary discretization

1. Click the **PREP_MESH** item of the general menu
2. Click the **NB_INTERVAL** item of the application menu (top)
3. Enter **4=** on the keyboard
4. ALL of the application menu (top) ???? All the elements are divided into 4 intervals
5. Enter **12=** on the keyboard
6. Then click in the graphic window close to the arc. It is divided into 12 intervals. There is no need to change selection mode.
7. Click the **SAVE** item of the general menu
8. Enter the file name **square_with_hole(CR)**. We have created the file *square_with_hole.emc2_bd*

The file `square_with_hole.emc2_bd` :

```
'-- TYPE N BD(1) BD(2) BD(3) BD(4) BD(5) NBNODE RATIO NUREFG NUREFD NUREF1G NUREF1D NUREF2G NUREF2D FIS'
'DROITE'       1 -1.000000 0.0000000 1.000000 0.0000000 0.0000000 0.0000000 0 0.0000000 0 0 0 0 0 0 F
'DROITE'       2 -1.000000 1.000000 0.0000000 0.0000000 0.0000000 0.0000000 0 0.0000000 0 0 0 0 0 0 F
'POINT'      3 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0 1.000000 0 0 0 0 0 0 F
'POINT'      4 0.0000000 1.000000 0.0000000 0.0000000 0.0000000 0.0000000 0 1.000000 0 0 0 0 0 0 F
'POINT'      5 0.0000000 0.0000000 1.000000 0.0000000 0.0000000 0.0000000 0 1.000000 0 0 0 0 0 0 F
'POINT'      6 0.0000000 1.000000 1.000000 0.0000000 0.0000000 0.0000000 0 1.000000 0 0 0 0 0 0 F
'POINT'      7 0.0000000 0.5000000 0.5000000 0.0000000 0.0000000 0.0000000 0 1.000000 0 0 0 0 0 0 F
'SEGMENT'    8 -3.000000 0.0000000 0.0000000 1.000000 0.0000000 0.0000000 5 1.000000 0 0 0 0 0 0 F
'SEGMENT'    9 -3.000000 1.000000 0.0000000 1.000000 1.000000 0.0000000 5 1.000000 0 0 0 0 0 0 F
'SEGMENT'    10 -3.000000 1.000000 1.000000 0.0000000 1.000000 0.0000000 5 1.000000 0 0 0 0 0 0 F
'SEGMENT'    11 -3.000000 0.0000000 1.000000 0.0000000 0.0000000 0.0000000 5 1.000000 0 0 0 0 0 0 F
'CERCLE'     12 0.2500000 0.5000000 0.5000000 0.0000000 0.0000000 0.0000000 2 1.000000 0 0 0 0 0 0 F
'ARC'        13 -2.000000 0.5000000 0.5000000 0.7500000 0.5000000 6.283185 13 1.000000 0 0 0 0 0 0 F
'MASQUE' 0 -0.4632312 1.465731 -5.0000010E-02 1.052500 0.0000000 0.0000000 0 0. 0 0 0  0 0 0  F
'RADIUS' 0 0.2500000 0. 0. 0. 0. 0. 0 0. 0 0 0  0 0 0  F
'ANGLE' 0 0.0000000 0. 0. 0. 0. 0. 0 0. 0 0 0  0 0 0  F
'DISTANCE' 0 1.000000 0. 0. 0. 0. 0. 0 0. 0 0 0  0 0 0  F
'RAISON' 0 1.000000 0. 0. 0. 0. 0. 0 0. 0 0 0  0 0 0  F
```

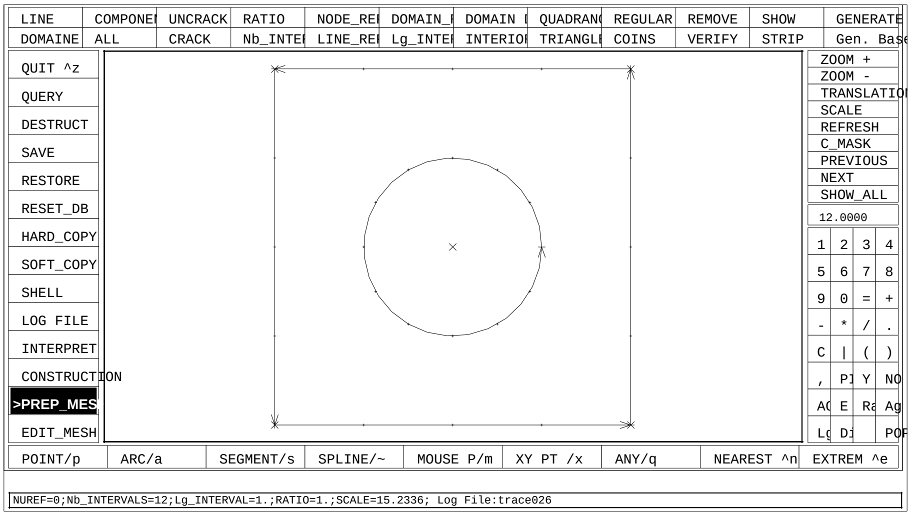

Figure 7.2: Discretization of the lines of the square domain with a hole

```
'NOMBRE' 0 0.0000000 0. 0. 0. 0. 0. 0 0. 0 0 0  0 0 0  F
'ECHELLE' 0 14.32432 0. 0. 0. 0. 0. 0 0. 0 0 0  0 0 0  F
'NUREF' 0 0.0000000 0. 0. 0. 0. 0. 0 0. 0 0 0  0 0 0  F
'NB_INTERVALS' 0 12.00000 0. 0. 0. 0. 0. 0 0. 0 0 0  0 0 0  F
'NUDSD' 0 1.000000 0. 0. 0. 0. 0. 0 0. 0 0 0  0 0 0  F
```

We have completed the preparation of the mesh (figure 7.2)

7.1.3 Generating and saving the mesh

1. Click the **`EDIT_MESH`** item of the general menu to activate the EDIT_MESH application. Then type 4 `(CR)` to select the default options of the mesh generator. The mesh is displayed, it includes the hole. We are going to suppress the mesh in the hole.

2. Click the **`S_DOM`** item of the selection menu.

3. Click the **`SUPPRESS`** item of the edit_mesh menu (top).

4. Click in the meshed circle. The mesh of the hole disappears.

5. Click the **`SAVE`** item then enter the type of format, for example **`am_fmt(CR)`**, then the file name **`square_with_hole(CR)`**. A file titled square_with_hole.am_fmt is generated.

6. Click the **`EXIT`** item to quit Emc^2

The file **`square_with_hole.am_fmt`**:

```
 44 60   -- nbs,nbt
      1       2      29          25      24      29
     24      31      29           3      43      30
     15      16      31          22      34      23
     26      25      30          23      34      31
     32      27      43          28      27      32
     22      33      34          21      40      38
      4      32      43          34      15      31
     33      14      34          21      33      22
      5       6      39          20      42      40
     28      32      39          41      42      19
      7       8      36          18      36      37
     28      39      35          34      14      15
     12      13      38           6      35      39
     28      35      17          43      27      26
     38      13      14          23      31      24
      7      36      17          31      16      29
     11      12      40           3      30       2
```

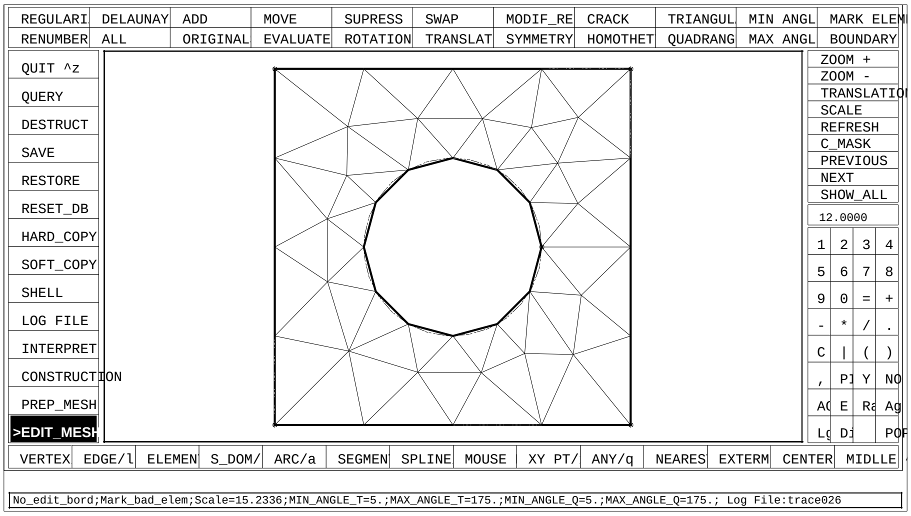

Figure 7.3: The final mesh in the square domain with a hole

```
  37       8      44          30      25      29
  41      10      42          30      29       2
  44      10      41          29      16       1
  35       6       7          17      35       7
  36       8      37          17      36      18
  41      37      44          18      37      19
  38      14      33          21      38      33
  39      32       4           5      39       4
  40      12      38          20      40      21
  42      10      11          37      41      19
  42      11      40          19      42      20
  43      26      30           4      43       3
  44       8       9          10      44       9
0.000000E+00   0.000000E+00     2.500000E-01   0.000000E+00
5.000000E-01   0.000000E+00     7.500000E-01   0.000000E+00
1.000000E+00   0.000000E+00     1.000000E+00   2.500000E-01
1.000000E+00   5.000000E-01     1.000000E+00   7.500000E-01
1.000000E+00   1.000000E+00     7.500000E-01   1.000000E+00
5.000000E-01   1.000000E+00     2.500000E-01   1.000000E+00
0.000000E+00   1.000000E+00     0.000000E+00   7.500000E-01
0.000000E+00   5.000000E-01     0.000000E+00   2.500000E-01
7.500000E-01   5.000000E-01     7.165064E-01   6.250000E-01
6.250000E-01   7.165064E-01     5.000000E-01   7.500000E-01
3.750000E-01   7.165064E-01     2.834937E-01   6.250000E-01
2.500000E-01   5.000001E-01     2.834936E-01   3.750001E-01
3.749999E-01   2.834937E-01     4.999999E-01   2.500000E-01
6.249999E-01   2.834936E-01     7.165062E-01   3.749999E-01
2.077567E-01   2.075703E-01     4.017089E-01   1.482612E-01
1.483122E-01   4.021604E-01     7.024530E-01   2.009149E-01
2.021900E-01   7.009841E-01     1.471843E-01   5.790281E-01
8.609530E-01   3.646002E-01     8.519265E-01   6.222693E-01
7.943417E-01   7.355210E-01     2.050139E-01   8.383228E-01
8.379703E-01   1.980158E-01     4.015485E-01   8.609668E-01
7.210978E-01   8.353896E-01     5.831956E-01   8.603261E-01
5.796704E-01   1.470356E-01     8.526899E-01   8.640727E-01
     0       0       0       0       0       0       0       0       0       0
     0       0       0       0       0       0       0       0       0       0
     0       0       0       0       0       0       0       0       0       0
     0       0       0       0       0       0       0       0       0       0
     0       0       0       0       0       0       0       0       0       0
     0       0       0       0       0       0       0       0       0       0
     0       0       0       0       0       0       0       0       0       0
     0       0       0       0       0       0       0       0       0       0
     0       0       0       0       0       0       0       0       0       0
     0       0       0       0       0       0       0       0       0       0
     0       0       0       0
```

7.2 A NACA0012 wing

7.2.1 Introduction

The parametrization of the naca0012 is

$$Y(X) = 5 \times 0.12(0.2969\sqrt{X} - 0.126X - 0.3516X^2 + 0.2843X^3 - 0.1015X^4)$$

for $X \in [0, 1.008930411365]$

To normalize the naca0012, we use the change of variables:

$$X = 1.008930411365x$$

$$Y = 1.008930411365y$$

With the following small fortran program:

```
      PROGRAM NACA12
      DOUBLE PRECISION X,Y,C,XX,YY
      PARAMETER (C=1.008930411365D0)
      INTEGER NB,I
C --------  ENTER THE NUMBER OF INTERVALS -----------
      READ *,NB
      DO I=0,NB
C ---------------------------------------------------
C              4                                   |
C       WE USE X  INSTEAD OF X SO THAT THE PROGRESSION  |
C       OF GENERATED POINTS IS DENSER AROUND 0          |
C ---------------------------------------------------
       XX= (DBLE(I)/DBLE(NB))**4
       X = C*XX
       Y = 5*.12*( 0.2969*SQRT(X)  -0.126*X
     +            -0.3516*X**2    +0.2843*X**3  -0.1015*X**4)
       YY = Y / C
       PRINT*, XX,YY
      ENDDO
      END
```

We generate the file "naca12.21pts" by using this program with 20 intervals.

The file "naca12.21pts":

```
0.0000000000000000 0.0000000000000000
6.2500000000000000E-06 4.4290213310617220E-04
1.0000000000000000E-04 1.7659364374154260E-03
5.0625000000000000E-04 3.9520447457293480E-03
1.6000000000000000E-03 6.9724900930098020E-03
3.9062500000000000E-03 1.0785816129632120E-02
8.1000000000000000E-03 1.5335254409279880E-02
1.5006250000000000E-02 2.0543538719326360E-02
2.5600000000000000E-02 2.6304013971036800E-02
4.1006250000000000E-02 3.2467169756843240E-02
6.2500000000000000E-02 3.8822480697810590E-02
9.1506250000000000E-02 4.5076893778196040E-02
0.1296000000000000 5.0833559338836810E-02
0.1785062500000000 5.5577241169031020E-02
0.2401000000000000 5.8675384962268570E-02
0.3164062500000000 5.9403968668221290E-02
0.4096000000000000 5.7000809116986590E-02
0.5220062500000000 5.0728415111155510E-02
0.6561000000000000 3.9881091560527040E-02
0.8145062500000000 2.3576580331835940E-02
1.000000000000000 2.0060198232592660E-14
```

7.2.2 Constructing the geometry

1. Click the `SPLINE` item of the construction menu
2. Enter `<` on the keyboard or click the `XY FILE` item of the selection menu then enter the name of the file that contains the points of the NACA0012 by typing `naca12.21pts(CR)`. *The spline of the naca appears on the screen.*
3. Click the `SEE_ALL` item to see the spline clearly.

4. We want to mesh the exterior of the naca0012. To do this, we must define the boundary at infinity that we will place at -5 upstream, +11 downstream and +8 in the orthogonal direction. We thus construct the 3 points (-5,0), (0,8), (11,0)

   - Click the `POINT` item of the construction menu (top)
   - Enter `x-5=0=0=8=11=0=` on the keyboard
   - Click the `SEE_ALL` item (right)

   The 3 points appear. We now construct the arc passing through these 3 points:

   - Click the `ARC` item of the construction menu (top)
   - Click the `POINT` item of the selection menu (bottom)
   - Click successively near the points (11,0), (0,8), (-5,0)

5. To finish the construction of the domain, we must construct the two segments that go from "infinity" to the naca.

   - Click the `SEGMENT` item of the construction menu (top)
   - Click near point (-5,0)
   - Click the `PREVIOUS` item of the screen management menu (right). We are now using the previous display context that sets the naca full screen.
   - Click near point (0,0) (below and on the left, to be sure to select the first point).
   - Click the `NEXT` item of the screen management menu (right). *We now see all the objects.*
   - Click near point (11,0)

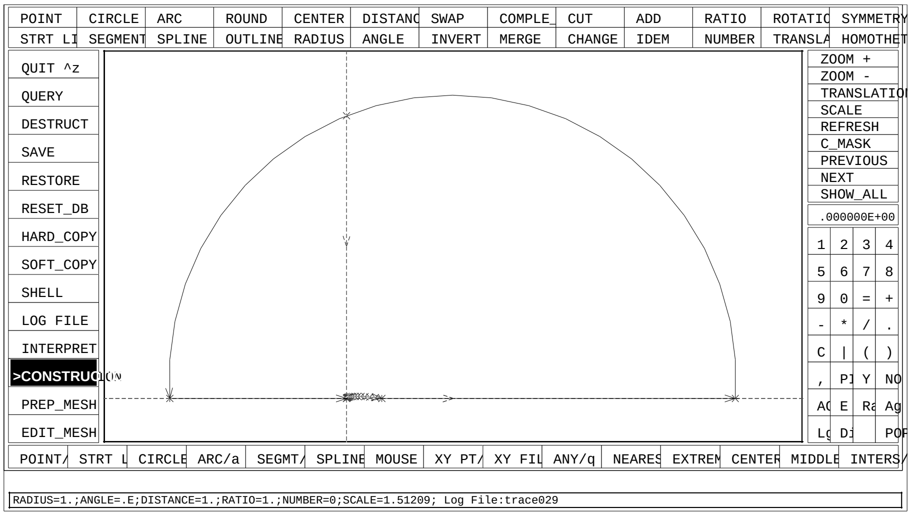

Figure 7.4: The complete geometry of the naca0012 and infinity

The construction of the geometrical domain is done. (figures 7.4 and 7.5).

7.2.3 Discretization of the contours

We switch to the PREP_MESH application to define the discretization of the contours.

1. Click the `PREP_MESH` item
2. The discretization of lines is done as follows:

   - Click the `NB_INTERVAL` item. *We plan to define the number of intervals.*

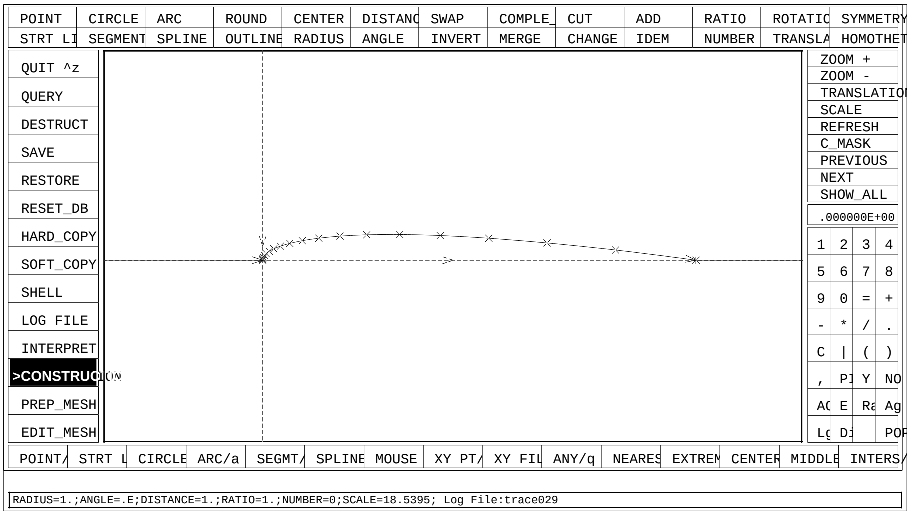

Figure 7.5: Zoom around the naca0012

- Type `26=`
- Click near the arc (Warning: we are in the ANY selection mode, thus we must not click near point (8,0)). The arc is divided into 26 intervals.
- Type 39=
- Click near the left segment.
- Type `30=`.
- Click near the right segment.
- Type `50=`.
- Click the **`SPLINE`** item of the selection menu.
- Click in the graphic window (there is only one spline which is thus selected).
- Click the **`RATIO`** item *to change the repartition of points on the lines*
- Type `1.1=`
- Click near the spline.
- Click the **`SEGMENT`** item of the selection menu.
- Click near the right segment.
- Type `1/1.2=`
- Click near the left segment.

The discretization of lines is done (figures 7.6 and 7.7).

7.2.4 Generating and editing the mesh

We generate the mesh by switching to the EDIT_MESH application.

1. Click the **`EDIT_MESH`** item then type `(CR)` 4 times to give the default answer to the 4 questions. A mesh appears (figure 7.8).

   We now define references in order to take boundary conditions into account: 1 on infinity Γ_∞, 2 on the NACA, 3 on the trailing edge.

2. Click the **`MODIF_REF`** item of the application menu.

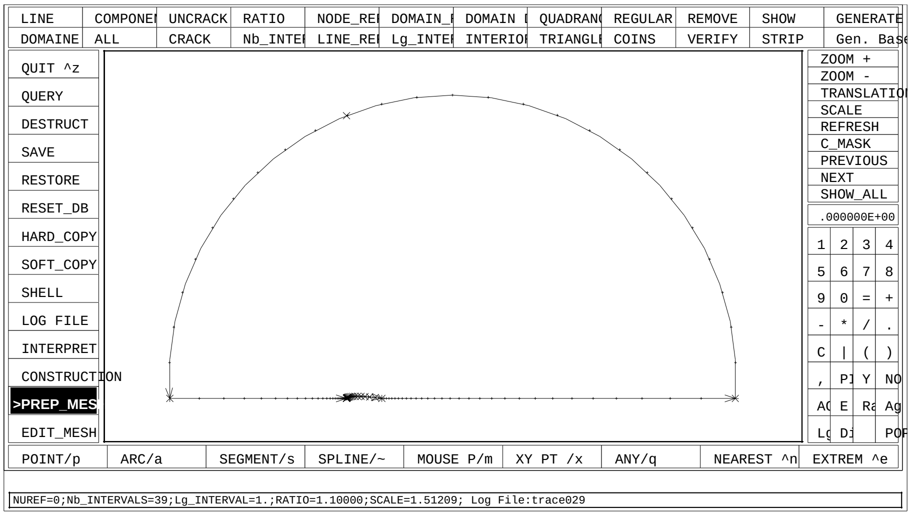

Figure 7.6: A view of the discretization of all contours

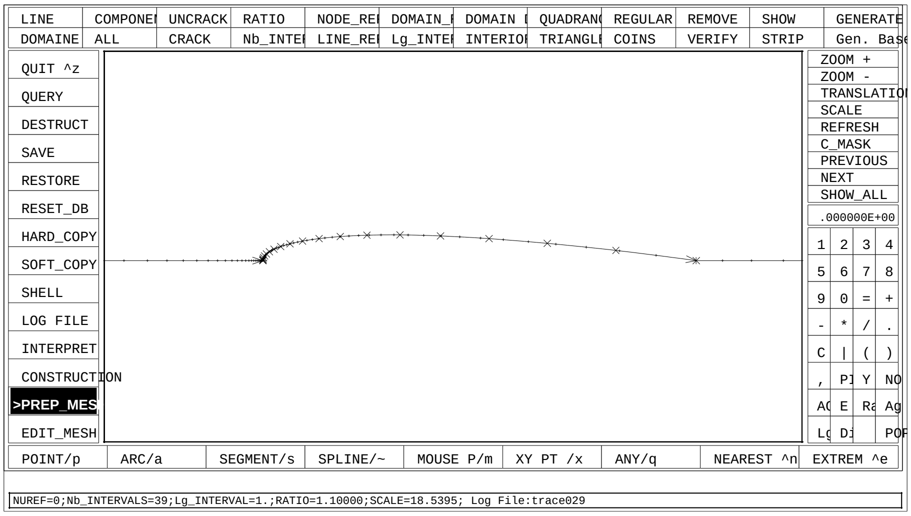

Figure 7.7: Zoom on the discretization of the contours around the naca

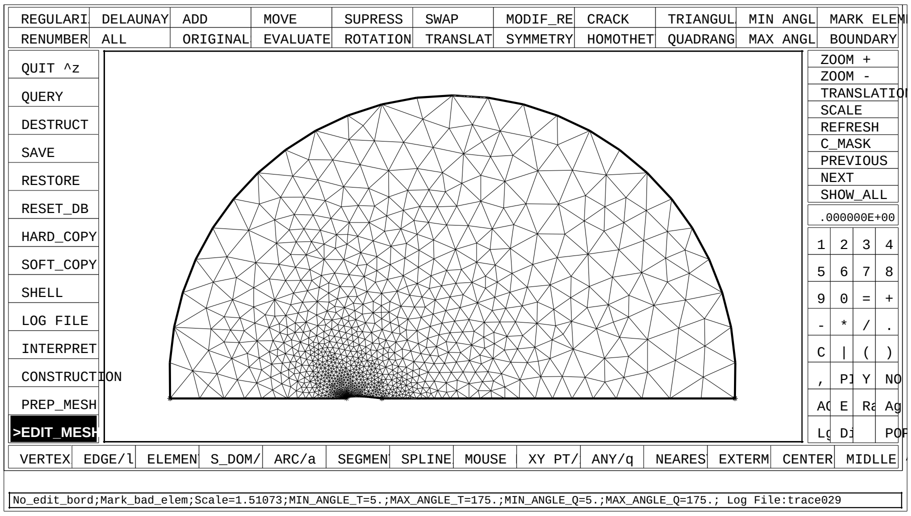

Figure 7.8: Mesh around the half naca0012

- Type `1=` *The current reference is 1*
- Click the `VERTEX` item of the selection menu to switch to vertex selection mode
- Click in the mesh near a vertex that is on the arc but is not an extremity, then click in the mesh near the two extremities. *All the vertices on the arc have reference 1.*
- Type `2=` *The current reference is 2*
- We zoom around the naca
  - Click the `ZOOM +` item of the screen management menu
  - Click the `MOUSE P` item of the selection menu to switch to mouse point selection mode
  - Click twice to select the two opposite corners of a fictitious rectangle that contains the naca. If the zoom is not sufficient, click the `ZOOM +` item again, then click two other opposite corners. In order to get a zoom around the beginning of the naca (coordinates (0,0)):
    - Click the `C_MASK` item then type `x0=0=` to center the graphic window on point (0,0) (x is the shortcut for `XY POINT`)
    - Click the `SCALE` item then type `1000=` for a scale of 1000

  (figure 7.9)
- Click the `VERTEX` item of the selection menu to switch back to vertex selection mode
- In case of error—the MODIF_REF item is not marked anymore—click it again to continue and type `2=` to redefine the current reference.
- Click in the mesh near a vertex that is on the naca but is not an extremity, then click in the mesh near the downstream extremity (coordinates (0,0)).
- Click the `C_MASK` item then type `x0=1=` to center the graphic window on point (0,1) which is the trailing edge.
- Type `3=` *The current reference is 3*
- Click in the mesh near the trailing edge.

Entry of references is done. Now we save the mesh of the half naca.

3. Click the `SAVE` item then type `mesh(CR)` then `naca12_5(CR)`. The file *naca12_5.mesh* is created.

To obtain a complete mesh, we make it symmetrical. First we change point of view to see the whole naca.

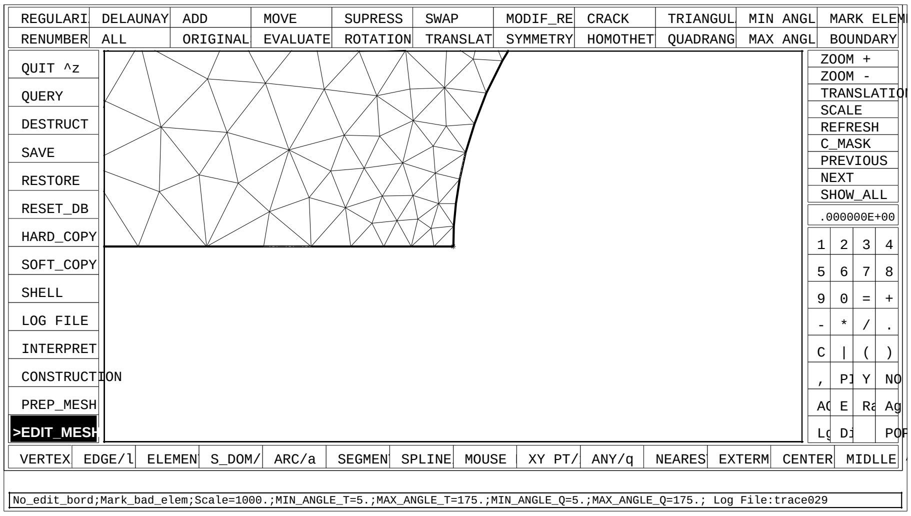

Figure 7.9: Zoom 1000 on the mesh of the half naca0012

4. Click the **C_MASK** item then type **x1/2=0=** *to center the graphic window on point* $(\frac{1}{2}, 0)$
5. Click the **SCALE** item then type **10=** for a scale of 10
6. Click the **SYMMETRY** item
   - Click the **SEGMENT** item of the selection menu.
   - Click near one of the 2 segments
   - Click the **ALL** item. *The whole mesh is made symmetrical. The symmetric part is displayed in dotted lines (figure 7.10).*

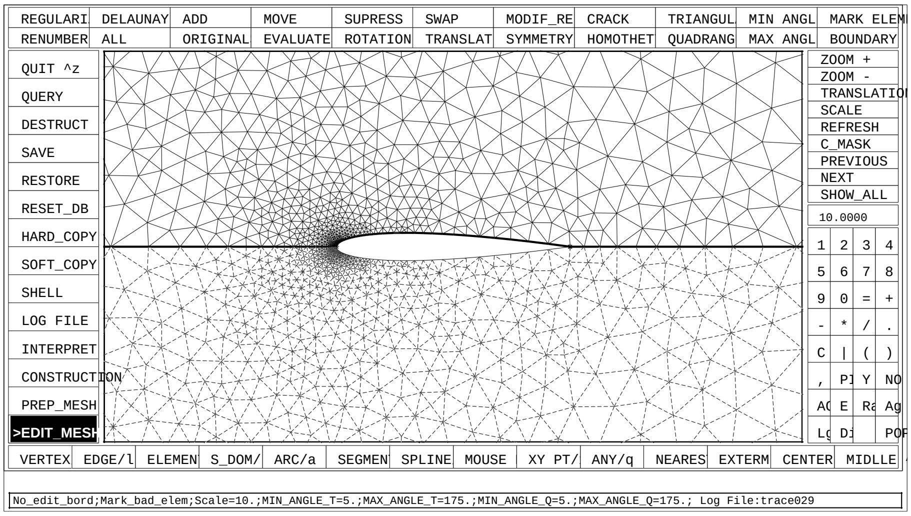

Figure 7.10: Zoom 10 on the symmetric mesh of the naca0012

   - Click the **EVALUATE** item to effectively generate the entire mesh (figure 7.11).

Now we save the mesh of the naca.

7. Click the **SAVE** item then type **mesh(CR)** then **naca12(CR)**. The file *naca12.mesh* is created.

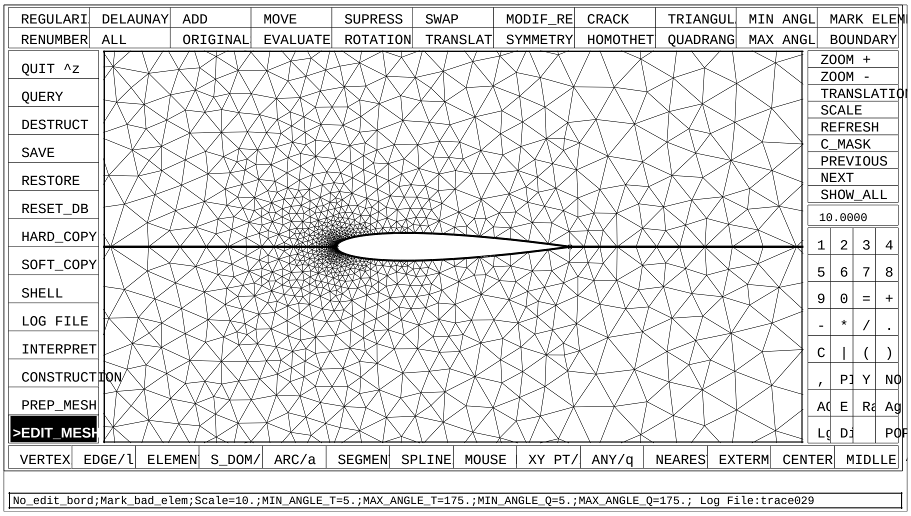

Figure 7.11: Zoom 10 on the entire mesh of the naca0012

Chapter 8

Limitations and bugs

- Files that can be created by the system
  If xx is the name given by the user, the file names that can be created from xx are:

traceNNN.emc2_trace generated names).	Trace of operations (automatically
xx.emc2_trace	Trace of operations (name given by the user).
xx.emc2_bd PREP MESH	Save under CONSTRUCTION or under
xx_bak.emc2_bd	.bak file of the previous one.
xx.data	File for the APNOPO module (it will generate xx.nopo).
xx.nopo	File generated by EDIT MESH.
xx.mesh	File generated by EDIT MESH.
xx.am	File generated by EDIT MESH.
xx.am_fmt	File generated by EDIT MESH.
xx.ambda	File generated by EDIT MESH.
xx.set	For linking with the VISIL software.
xx.bas	For linking with the VISIL software.

- Initialization of global variables
  Maximum number of
  - DB elements MXBDX=10000
  - discretization points on a DB element MXNOD=2000
  - points defining a spline MXPDEF=200
  - vertices in a mesh MBPMXX=50000
  - triangles in a mesh NBTMXX=2*NBPMXX-2=99998
  - edges that define the boundaries in a mesh NBAMXX=10000
  - subdomains MBSDMXX=2000
  - list elements MXLISTX=50000

  All these variables are initialized by the `emc2.f` program (`ppal` directory), except MXNOD and MXPDEF, which are initialized in the include file `emc2_commons_g.ins include` directory).
- We must be careful when several DB elements are superimposed, for example a segment and its supporting line, when the selection mode is ANY: the result of the selection is unpredictable.
- When switching from the CONSTRUCTION application to the PREP_MESH application, if two arcs, segments or splines are superimposed, one of them is destroyed.

- If the mesh generator, which is called when switching to the EDIT_MESH application, refuses to construct the mesh, the reason is in all probability that:
  - **all segments, arcs and splines must be cut explicitly** i.e. an extremity of a segment, arc or spline cannot lie on a curve that is not cut.
  - when we construct arcs with properties of tangency, we can encounter rounding off error problems. To avoid this, we must construct the points of tangency first and then use them.
  - the precision of the mesh generator is 1/32000 with respect to the larger length. If we ask for a too fine discretization, the mesh generator can either turn triangles over or refuse to mesh, because some points are considered equal.
  - a curve is not discretized enough: the boundary is (self?) crossed.

  the type of error is: some points are considered equal or the boundary is (self?) crossed (or crosses itself?)
- Generally, the mesh generator will not conserve the symmetries of the domain. If we want to keep these symmetries, we must first mesh the minimal part, then generate the entire mesh by using the transformations of the EDIT_MESH application.
- Warning: using transformations of the EDIT_MESH application may result in the construction of overlapping subdomains. The results are unpredictable.
- Warning: the mesh points that are read by the EDIT_MESH application are not appended to the points of already existing meshes.
- Warning: all the modifications of a mesh are lost when exiting the EDIT_MESH application. The mesh is lost and only the contours are saved.
- Warning: because of a fortran problem (INQUIRE), it is impossible to execute emc2 in a write protected directory (the programme attempts to open 999 `LOG FILE` files, see 3.3.10, which takes some time)
- If you lose information in the graphic window, the only way to refresh it is to resize it and then click inside.

Chapter 9

Internal data structure

We use a system of lists with two elements, which we call CAR and CDR. The empty address is NIL=0 (parameter). In the following pictures, CAR will always be on the left and CDR on the right.

CAR	CDR

The free atoms are in a list of free atoms. The beginning of this list is FREE. Free atoms are linked through their CDR.

9.1 Description of the DB for the CONSTRUCTION application

TYPE	1	2	3	4	5	NUREF D & G	NUREF1 D & G	NUREF2 D & G	NBNODE	RAISON	FISSUR T/F
POINT=0	X	Y				nuref			>=2	>0	
RAYON>0	XC	YC							>=2	>0	
DROITE	A	B	C						>=2	>0	
SEGMENT	X1	Y1	X2	Y2		nuref	nuref1	nuref2	>=2	>0	
ARC	XC	YC	X1	Y1	ANGLE	nuref	nuref1	nuref2	>= 2|3|4	>0	
SPLINE	NB	PT *				nuref	nuref1	nuref2	>=3|4	>0	

```
                    |
 |----------<-----| Liste des NB points de definition de la SPLINE
 v
 |------->|  *  |  *-------------->|  *  |  *----------->|  *  |  nil  |
             |                        |                    |
             v                        v                    v
       point en B.D.            point en B.D.        point en B.D.
```

Figure 9.1: Description of the DB for the CONSTRUCTION application

Figure 9.1 describes the DB for the CONSTRUCTION application with,

TYPE the type of DB element

- POINT = 0. X,Y POINT.
- RADIUS > 0. CIRCLE of center XC,YC and radius RADIUS.

- LINE = -1. STRAIGHT LINE of equation $A*x+B*y+C=0$ with $A*A+B*B=1$
- SEGMENT = -3. SEGMENT going from point X1,Y1 to point X2,Y2.
- ARC = -2. ARC of center XC,YC passing through point X1,Y1 and with an angle ANGLE from this point.
- SPLINE = -4. SPLINE.
- EMPTY = -1000. empty element.

for SPLINE,

- NB is the number of definition points.
- PT is the pointer on the list of definition POINTS of the SPLINE.
- NUREF is the reference number of the element (ARC or SEGMENT). Default: NUREF=0.
- NUREF1 is the reference number of extremity 1 of the element (ARC or SEGMENT). Default: NUREF1=0.
- NUREF2 is the reference number of extremity 2 of the element (ARC or SEGMENT). Default: NUREF2=0.
- NBNODE is the number of intermediate points in the element (extremities are included). Default: NBNODE=2.
- RATIO is the ratio of the geometric progression used to distribute the NBNODE-2 points from extremity 1 to extremity 2. Default: RATIO=1.

9.2 Description of the DB for the PREP MESH application

Figure 9.2 describes the DB for the PREP MESH application, where

TYPE is the type of DB element

- POINT = 0. X,Y POINT.
- SEGMENT = -3. SEGMENT going from point X1,Y1 to point X2,Y2.
- ARC = -2. ARC of center XC,YC passing through point X1,Y1 and with an angle ANGLE from this point.
- SPLINE = -4. SPLINE.
- EMPTY = -1000. empty element.

NUREF is the reference number of the element (ARC or SEGMENT).

ADP1 DB pointer on the POINT extremity 1 of the element (If the element is not cracked, NUREF of this POINT contains the reference number of extremity 1 of the element).

ADP2 DB pointer on the POINT extremity 2 of the element or on the last POINT of the SPLINE. (If the element is not cracked, NUREF of this POINT contains the reference number of extremity 2 of the element).

NBNODE is the number of intermediate points in the element (extremities are included).

RATIO is the ratio of the geometric progression used to distribute the NBNODE-2 points from extremity 1 to extremity 2.

CONX(2) DB pointers, used for the circular chaining of the elements of a connex component. Given one element, there are two and only two connex component that contain this element: the left component and the right component (left and right are relative to the orientation of the element). There are thus two chainings denoted by CONXG and CONXD.

CNX(2) indicates which CONX is to be used for the next element (CONXG or CONXD) depending on whether CNX equals LEFT or RIGHT (LEFT=1, RIGHT=2)(parameter).

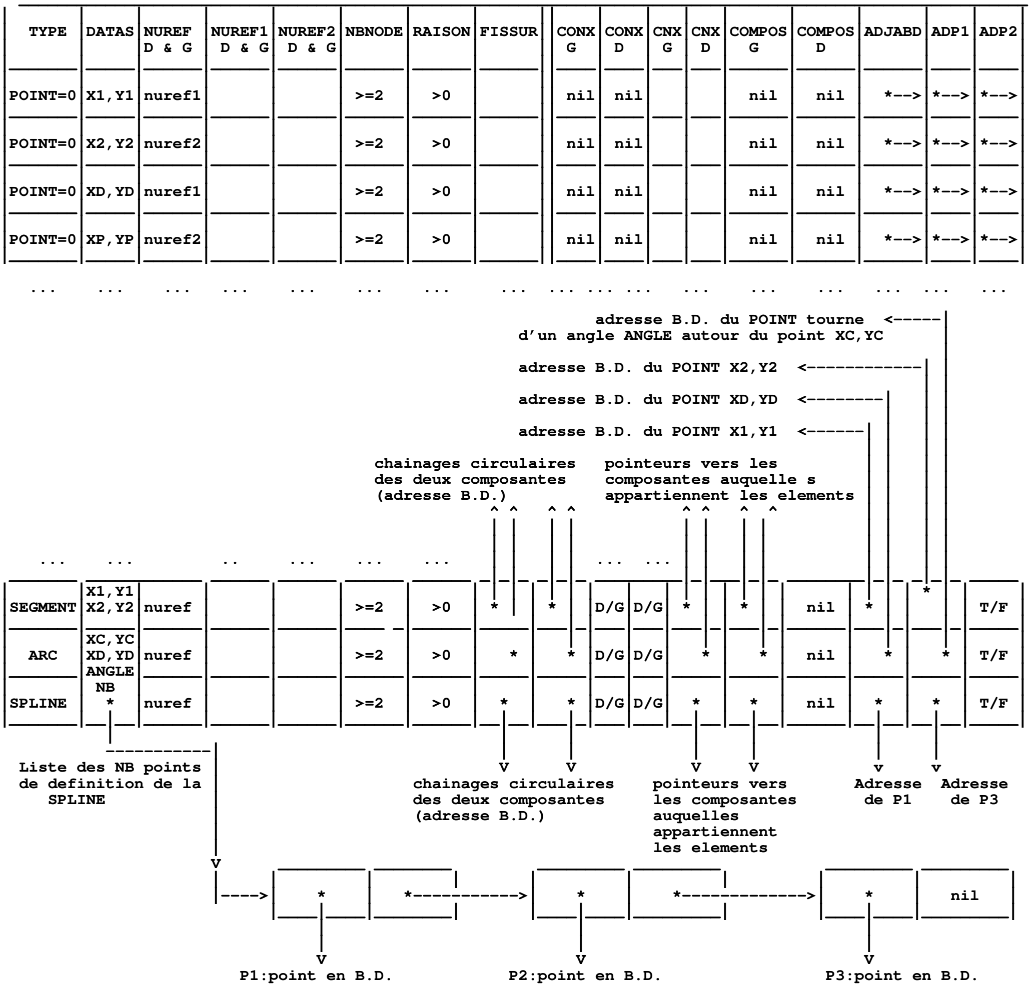

Figure 9.2: Description of the DB for the PREP MESH application

COMPOS(2) inverse pointers to the two components to which the element belongs. CNXG is associated with CONXG and COMPOSG (exist only for ARCS and SEGMENTS). CNXD is associated with CONXD and COMPOSD (exist only for ARCS and SEGMENTS).

ADJABD Beginning of the list of elements passing through the point (exists only for POINTS). (See the description of this list below).

9.2.1 Description of the list of elements passing through point i

The beginning of the list of elements passing through a point is ADJABD.
Figure 9.3 describes the list of elements passing through point i.

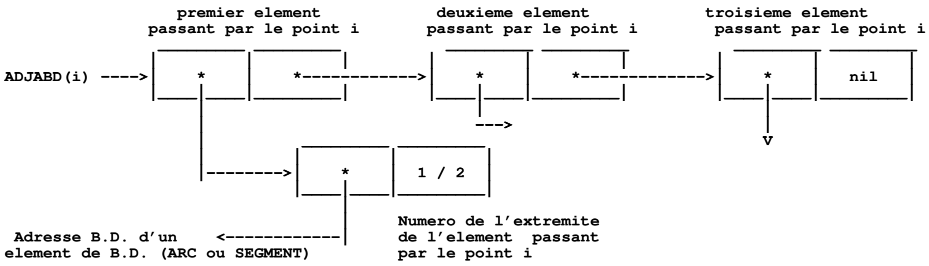

Figure 9.3: Description of the list of elements passing through point i

9.2.2 Description of the list of components

The beginning of the list of components is COMP.
Figure 9.4 describes the list of components.

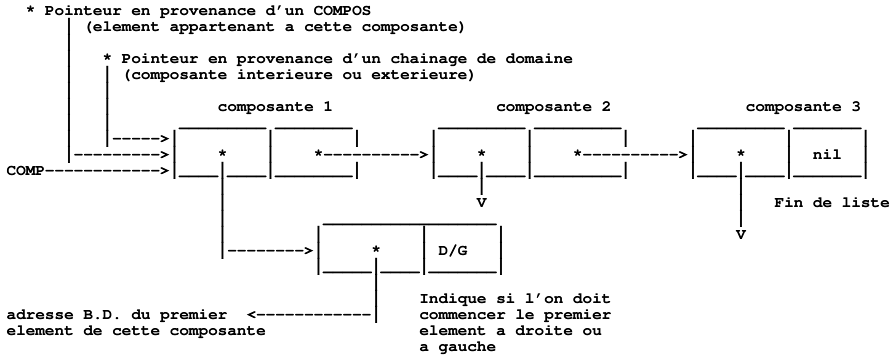

Figure 9.4: Description of the list of components

9.2.3 Description of the list of domains

The beginning of the list of domains is SDOMN.
Figure 9.5 describes the list of domains.

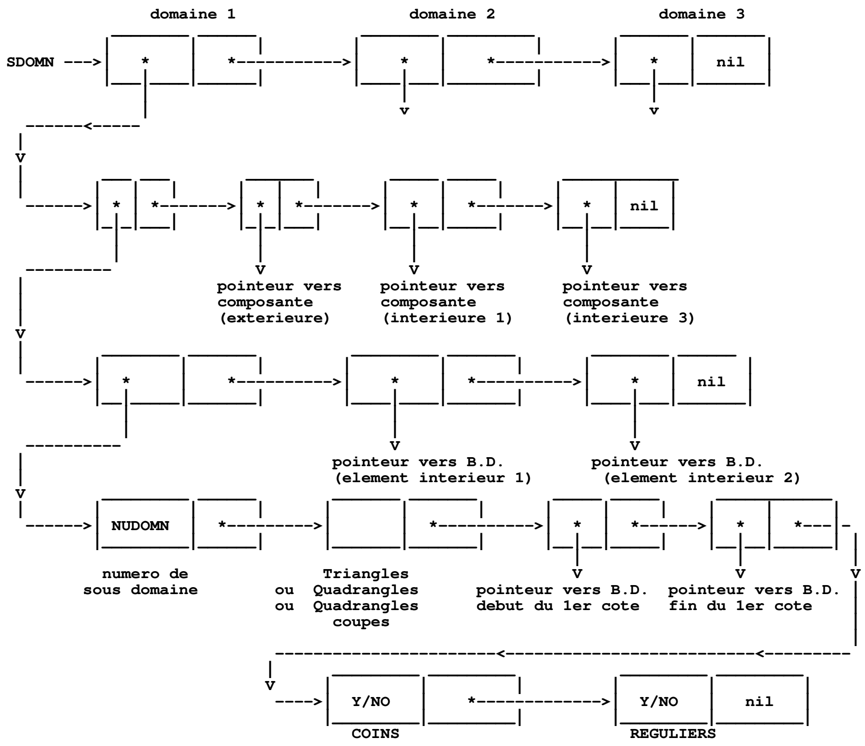

Figure 9.5: Description of the list of domains

9.3 Description of the DB for the EDIT_MESH application

The DB is the same as in the PREP_MESH application and the data structure DS_MESH is as follows:

nbs is the number of the last vertex

nbt is the number of the last triangle

nba is the number of edges that lie on curves (segment, arc, spline) that we will call boundary edges

nbsd is the number of subdomains

nbsrft is the number of vertices that are referenced by the elements

nbtria is the number of triangles

nbquad is the number of quadrangles

nbtrou is the number of holes in the mesh

freetr is the beginning of the list of detroyed triangles

finbd3 is the number of the last DB element of PREP_MESH

The vertices are defined in the following four parallel arrays:

cr(1:2,nbs) is the array of vertex coordinates.

nsorig(1:nbs) is the array giving the original vertex number of vertices that lie on a crack. The actual coordinates of vertex i are $cr(1:2, nsorig(i))$, while coordinates $cr(1:2,i)$ are only used to visualize the crack.

Note: if vertex i is not on a crack, then $nsorig(i) = i$.

abcurv(1:nbs) is:

- 0 if the vertex is not on a curve (segment, arc, spline), i.e. is an internal vertex of a subdomain,
- the abscissa of the point on a curve if this vertex is not an extremity of the curve, 0 if this vertex is the extremity of a curve. *Note: curves are always parametrized between* 0 *and* 1.

refs(1:nbs) is the array giving the DB number of the element (point, segment, arc, spline) that supports the vertex, if it exists, or 0 otherwise.

Note: if the vertex is the extremity of a curve then the element in question will be the point extremity of the curve (cf. ADP1 and ADP2 (Appendix 9.2))

The finite elements are triangles or quadrangles, which are internally represented with triangles whose edges are visible or not. This is used to draw the quadrangles. The arrays that define the triangles are:

nsea(1:6,nbt) gives for each triangle the vertices and the adjacent triangles or boundaries edges. If $nsea(1,ie) < 1$ then triangle ie does not exist and $reft(ie)$ gives the next destroyed triangle. Otherwise, triangle ie exists and

- $nsea(1:3,ie)$ gives for each triangle the vertex numbers counterclocwise.
- $d_i \stackrel{\text{def}}{=} nsea(i,ie), i = 4,5,6$, the 3 edges a_i of the triangle numbered from 4 to 6 have as vertices $nsorig(nsea(i-3,ie))$ and $(nsorig(nsea(mod(i,3)+1,ie))$.
  - if $d_i < 0$ the edge is a boundary edge. It is defined in $aretdb$ with number $-d_i$,
  - otherwise the edge is internal and $d_i = 8 * t_a + a_{t_a}$ where t_a is the number of the triangle adjacent to the edge and a_{t_a} is the number of the edge in t_a.

anovue(1:nbt) gives the visibility of the 3 edges for each triangle. If $anovue(i)$ equals

0 all the edges of triangle i are visible,

1 only edge 4 of triangle i is not visible,

2 only edge 5 of triangle i is not visible,

3 only edge 6 of triangle i is not visible.

reft(1:nbt) is:

- the array of the numbers of subdomain that contain the triangle,
- or the array of the chaining of triangles in a subdomain. $reft(ie)$ gives the next triangle in the subdomain,
  if $reft(ie) <> finsd = 2^{30}$, see also the $tetdt$ array for the beginnings of lists of subdomains.

The boundary edges of the mesh are defined as follows:

aretbd(1:2,nba) gives for each edge its two vertex numbers,

areadj(gauche:droite,nba) : $d_i \stackrel{\text{def}}{=} areadj(i,j), i = gauche, droite$ same definition as $nsea$,

refa(nba) gives the DB address of the support of the edge, which always exists by definition.

The subdomains are defined in the following arrays:

refsd(nbsd) gives the reference numbers of subdomains, if $refsd(i) = videsd = -2^{30}$ then the subdomain does not exist.

trfsd(3,3,i) gives the transformation applied to subdomain $ptorsd(i)$ to obtain this subdomain $i(i \in 1,..,,nbsd)$.

strfsd(i) gives the sign of the transformation (for example: -1 for a symmetry with respect to a straight line)

ptorsd(i) gives the number of the original subdomain (if $ptorsd(i) = i$ then subdomain i is not transformed *(original subdomain)*)

tetsd(nbsd) is an array that gives for each subdomain the first triangle of the list of triangles of this subdomain (cf. $reft$)

9.4 Description of AM, 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

nbs 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,nbs) is a real array giving the 2 coordinates of each vertex.

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

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

Note: the vertices that are not referenced by a triangle will be eliminated. In this case the vertex numbering will be compressed accordingly.

9.4.1 AM files

`xxx.am` files are read as follows:

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

close(1)
```

9.4.2 AM_FMT files

`xxx.am_fmt` files are read as follows:

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

9.4.3 AMDBA files

`xxx.amdba` files are read as follows:

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

Chapter 10

Index

Index

Bibliography

[1] M.Bernadou and All, MODULEF: Une bibliothéque modulaire d'éléments finis, INRIA, 1988.

[2] P.G.Ciarlet, The finite element method for elliptic problem, North Holland 1978.

[3] J.L.Coulomb, Maillages 2D et 3D. Experimentation de la triangulation de Delaunay, Conférence on Automated mesh generation and adpatation, Grenoble 1987.

[4] P.L George, MODULEF: Génération automatique de maillage, Collection didactique n^o 2, INRIA, 1988.

[5] P.L George, MODULEF: Construction et modification de maillages, RT n^{o}104 INRIA 1988.

[6] P.L.George, F.Hermeline, Maillage de Delaunay d'un polyèdre connexe en dimension d. Extention à un polydédre quelconque, CEA-N-90,1989.

[7] F.Hermeline, Triangulation automatique d'un polyèdre en dimension N, RAIRO numerical analysis vol 16, n^o 3 1982.

[8] A.Jameson, T.J.Baker, N.P.Weatherill, Calculation of inviscid transonic flow over a complete aircraft. AIAA 24th Aerospace Sciences Meeting, Reno Nevada, USA, 1986.

[9] R.Lohner, P.Parikh, Generation of 3-D unstructured grids by the advancing front method, AIAA 26th Aerospace Sciences meeting, Reno Nevada, USA, 1988.

[10] A.Perronnet, Tétraèdrisation d'un objet multi-materiaux ou de l'exterieur d'un objet, R n^o 88005, LAN 189,Université Paris 6, 1988.

[11] J.Y.Talon, Algorithmes d'amélioration de maillages pour éléments finis en 2 et 3 dimensions, Conférence on Automated mesh generation and adaptation, Grenoble 1987.

[12] J.F.Thompson, A general three dimensional elliptic grid generation system on a composite block-structure, Computer Methods in Applied Mechanics and Enginneering, Vol 64, 1987.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399