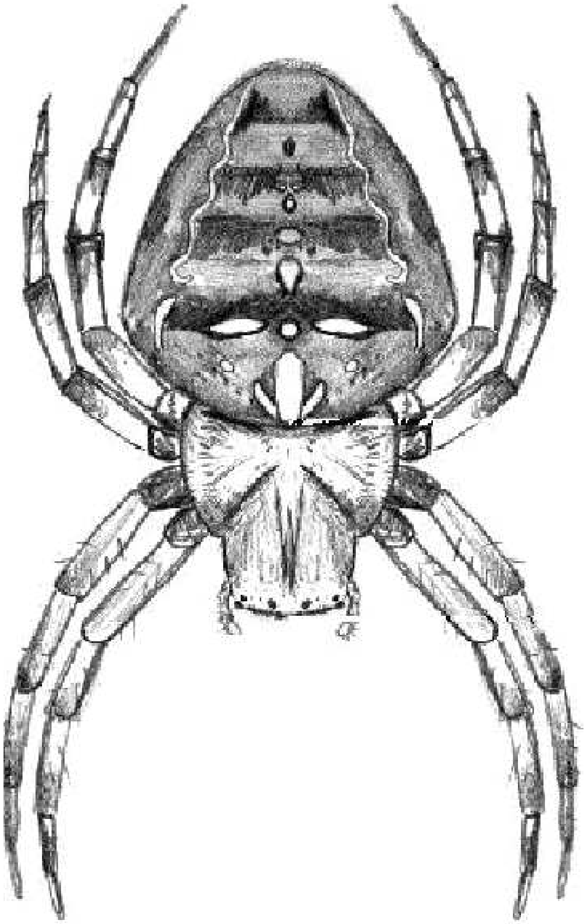
KumoNoSu

Version 1.0; March 24, 2008

USER'S MANUAL



Eric Hansen
Daniel Rypl
Victor Saouma
Department of Civil Engineering,
University of Colorado, Boulder
Boulder, CO 80309-0428

Under Contract from:
Tokyo Electric Power Service Company
3-3-3 Higashiueno, Taito-ku, Tokyo 110-0015

DISCLAIMER OF WARRANTIES AND LIMITATION OF LIABILITIES

THIS REPORT WAS PREPARED BY THE ORGANIZATION(S) NAMED BELOW AS AN ACCOUNT OF WORK SPONSORED OR COSPONSORED BY The TOKYO ELECTRIC POWER SERVICE COMPANY (TEPSCO). TEPSCO ANY COSPONSOR, THE ORGANIZATION(S) NAMED BELOW, NOR ANY PERSON ACTING ON BEHALF OF ANY OF THEM:

(A) MAKES ANY WARRANTY OR REPRESENTATION WHATSOEVER, EXPRESS OR IMPLIED, (I) WITH RESPECT TO THE USE OF ANY INFORMATION, APPARATUS, METHOD, PROCESS OR SIMILAR ITEM DISCLOSED IN THIS REPORT, INCLUDING MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, OR (II) THAT SUCH USE DOES NOT INFRINGE ON OR INTERFERE WITH PRIVATELY OWNED RIGHTS, INCLUDING ANY PARTY'S INTELLECTUAL PROPERTY, OR (III) THAT THIS REPORT IS SUITABLE TO ANY PARTICULAR USER'S CIRCUMSTANCES; OR

(B) ASSUMES RESPONSIBILITY FOR ANY DAMAGES OR OTHER LIABILITIES WHATSOEVER (INCLUDING ANY CONSEQUENTIAL DAMAGES, EVEN IF TEPSCO OR THEIR REPRESENTATIVES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES) RESULTING FROM YOUR SELECTION OR USE OF THIS REPORT OR ANY INFORMATION, APPARATUS, METHOD, PROCESS OR SIMILAR ITEM DISCLOSED IN THIS REPORT.

ORGANIZATION(S) THAT PREPARED THIS REPORT:

UNIVERSITY OF COLORADO AT BOULDER

ABSTRACT

KumoNoSu is a graphical front end to T3D (a finite element mesh generator), and T3D2Merlin (an structural analysis finite element analysis data preparation program).

KumoNoSu has been written explicitly for the Merlin finite element code, and as such has numerous built-in facilities to handle both cracks and/or dams. Nevertheless, the program is general enough to accommodate most other finite element codes.

KumoNoSu enables the user to interactively define the boundary of the structure to be meshed. Following each new boundary entity definition, the graphical display is updated. Once the boundary has been completely defined, then T3D is internally invoked, and a finite element mesh is generated. In the second module of KumoNoSu the user specifies the specific attributes of the finite element mesh (such as material properties, boundary conditions, incremental loads) with reference to the entity defining the boundary. This in turn will internally generate a complete input data file for the Merlin finite element code.

ACKNOWLEDGMENTS

KumoNoSu is a graphical front end to the mesh generator T3D and T3D2Merlin converter developed by Dr. Daniel Rypl.

KumoNoSu was originally developed by Dr. Eric Hansen, and subsequently modified/maintained by Dr. Gary Haussmann as a front end to the computer code MERLIN, through a contract from the Tokyo Electric Power Service Company (TEPSCO) with the Department of Civil Engineering of the University of Colorado in Boulder (Victor Saouma, P.I.).

The authors would like to acknowledge the numerous feedbacks, buggs reports and other constructive comments of: Guido Camata, Sonia Fortuna, C. Chang, Wiwat Puatsananon, and in particular of Takashi Shimpo and Yoshinori Yagome. It was through their numerous comments that KumoNoSu has matured into a solid, reliable, powerful and original program.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The finite element analysis requires the discretization of a structure into a mathematical representation using 1,2 or 3 dimensional elements.

The discretized structure is then subjected to the governing differential equation with essential (displacement) and natural (traction) boundary conditions.

A valid and well conditioned discretization of the structure being so important to the accuracy of the FEA solution that a mesh generator which produces a consistent, reproducible, high-quality mesh without great user intervention is key to a successful FEA.

Fig. 1.1 is a simple illustration of a physical problem, its boundary description, and the resulting finite element mesh.

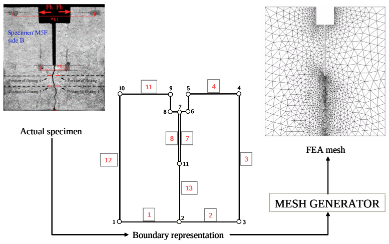


Figure 1.1: Mesh Generation Process

KumoNoSu is a graphical front end to two propgrams (both written by Dr. Daniel Rypl):

T3D a powerful mesh generator which can produce both unstructured (based on Delaunay triangularization) and structured meshes.

T3D2Merlin which enables definition of material properties, boundary conditions and loads for a Merlin input file.

Hence, KumoNoSu first produces a boundary definition of the physical object to be discretized, a tt .bd file. Once the boundaries of the solid are delineated and mesh-size generation guidelines established for the code, the program constructs a mesh to describe the structure, this is the `.t3d` file. Next, material properties and loading data must be added for each element to the geometric data set. This is saved into a `.ctrl` file. Finally, a finite element analysis algorithm may process the expanded data set into the Merlin `.inp` file, Fig. 1.2.

1.1 Concepts of Boundary Representation

The role of the boundary representation for T3D is as a geometrical description of individual model entities and representation of their topological relationships, Fig. 1.3:

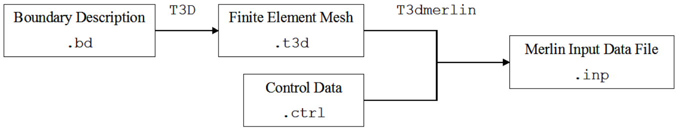


Figure 1.2: KumoNoSu 's File Types

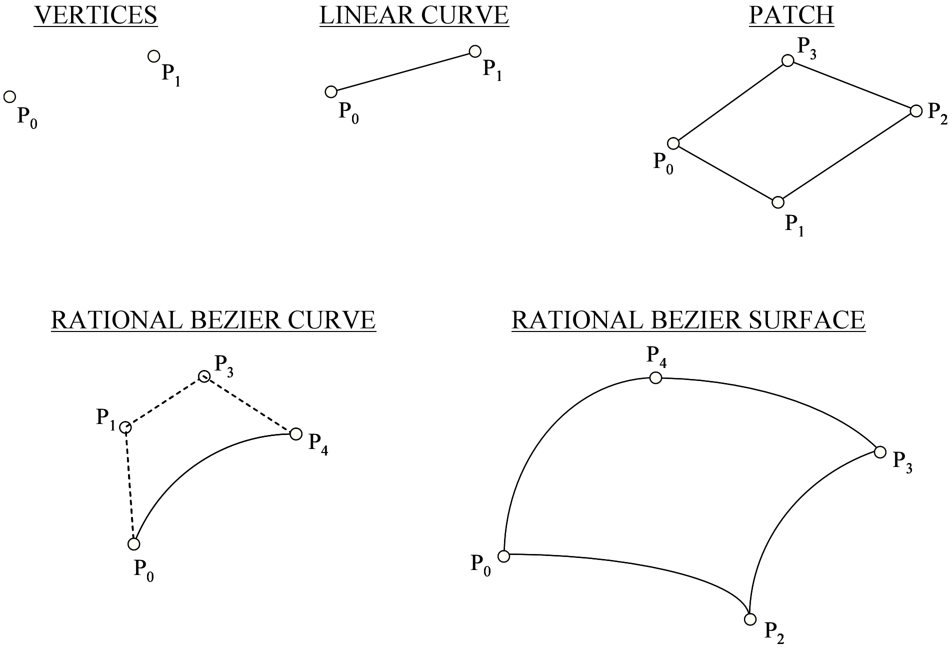


Figure 1.3: Entities Recognized by KumoNoSu

Vertices points in (x,y,z) space.

Curves defined by 2 end vertices, may be linear, quadratic, or cubic.

Patches planar collection of curves.

Surfaces non-planar, defined by (4) curves.

Shells non-planar collection of curves.

Regions set of non-self-intersecting boundary surfaces, patches, and shells.

1.1.1 Hierarchy

Those entities are defined hierarchically, Fig. 1.4 Lower level entities which belong to higher level entities are called the **Minor** entities of that higher level entity. Hence:

- Vertices are the minor entities of curves.
- Vertices and curves are the minor entities of patches and surfaces.
- Vertices, curves, surfaces, and patches are the minor entities of regions.

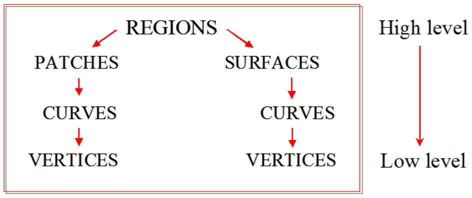


Figure 1.4: Hierarchy of Model Representation

1.1.2 Mesh Size/Density

In the process of generating a finite element mesh, it is highly desirable that one can control the mesh gradation or density. Hence, wherever the strain energy gradient is highest (such as in zones of stress concentration), we have a refined mesh. For unstructured meshes, this can be controlled by the **size** of an entity for triangular (2D) or tetrahedral (3D) meshes. In Fig. 1.5 we note how the reduction of the `size` parameter results in denser uniform mesh over the entire patch.

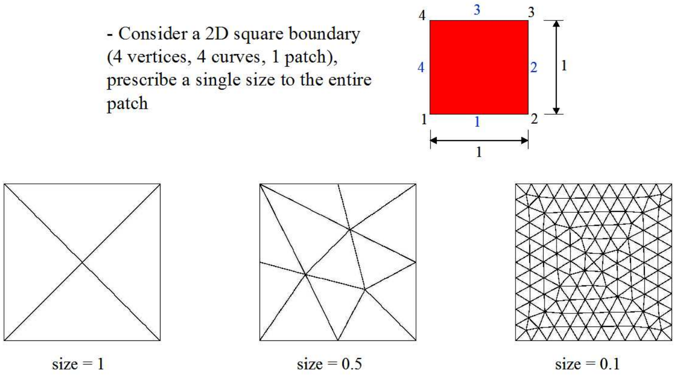

- Consider a 2D square boundary (4 vertices, 4 curves, 1 patch), prescribe a single size to the entire patch
4
3
3
4
2
1
1
1
2
1
size = 1
size = 0.5
size = 0.1

Figure 1.5: Concept of Mesh Size

For structured meshes, composed of quadrilaterals in 2D or hexagonal in 3D, mesh density can be controlled by the **count** concept, Fig. 1.6.

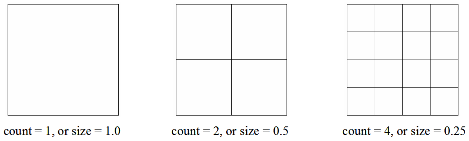


Figure 1.6: Concept of Mesh Count

Finally, **gradation** of the mesh can be accomplished by assigning different sizes to different entities (small size in the areas of denser mesh, larger size in areas of coarser mesh), Fig. 1.7.

1.2 Kumo Layout

KumoNoSu toolbars has the following main components, Fig. 1.9:

File Controls input/output with external files, Fig. 2.1. This is defined in described in 2

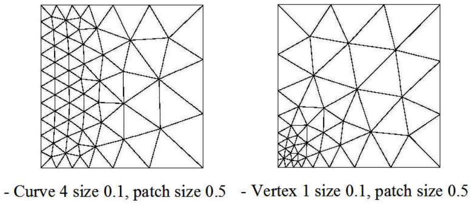

- Curve 4 size 0.1, patch size 0.5
- Vertex 1 size 0.1, patch size 0.5

Figure 1.7: Concept of Mesh Size; Curve and Patch

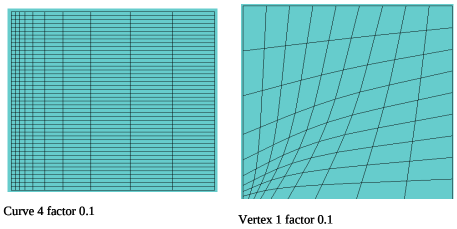


Figure 1.8: Concept of Factor

Define Boundary Allows user to define the boundary representation of the structure to be meshed, Chapter 3.

View Controls a number of viewing parameters, Chapter 4.2.

Generate Mesh to instruct `T3D` to generate the finite element mesh using the previously defined `.bd` file, Chapter **??**.

T3D2Merlin As the main interface which allows definition of the material properties, loads, boundary conditions, Chapter 6.

Help For setup and info about the code.

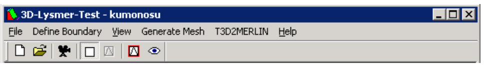


Figure 1.9: KumonoSu Toolbar

each one of them will be described in a separate chapter.

Chapter 2

File

The user clicks the file menu to initiate file management, Fig. 2.1 If the session is the initial session for the project, there is no need to select the `File` task.

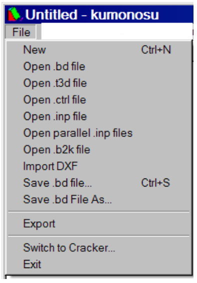


Figure 2.1: File Management

New Wipes out the memory, and enable the definition of a new bd file.

Open bd file If the user has previously created a boundary file and wishes to run another trial using the same boundary, the user should Open boundary file. KumoNoSu will present the user with the available `.bd`-files stored in the default directory. Even if stored in directories other than the default directory, other `.bd`-files are available by directory search, Fig. 2.2.

Open t3d file This will allow the user to open an existing mesh generated by `t3d`. The user may not modify the mesh defined in this file.

Open ctrl file This file contains the load and material properties information used by T3d2Merlin to generate a Merlin.inp file.

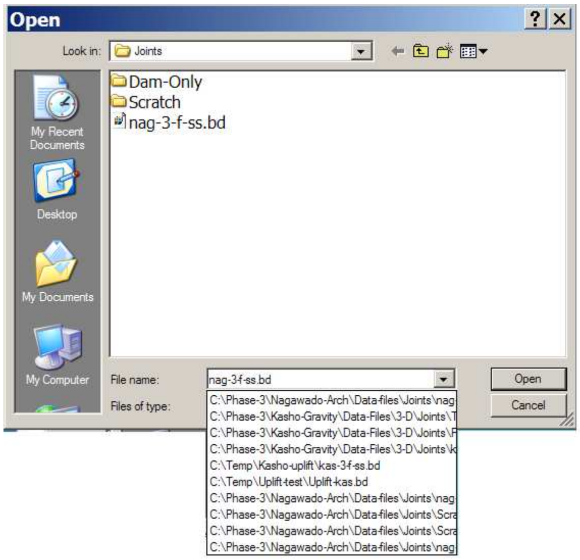
Open inp file to retrieve a Merlin file. There are few instances where this is needed.

Open Parallel input file Eanables the user to select one of the multiple files previously generated by KumoNoSu through its domain decomposition algortihm.

Open b2k file Opens a file coming from Beaver.

Import dxf file Import a dxf file (AutoCad). KumoNoSu will attempt to map vertices and curves. Surfaces and Regions (3D entities) are not recognized by KumoNoSu and will have to be manually defined.

Save bd file Once the boundary has been named, the user may save the entity at any phase of development.



Look in: Joints

Dam-Only
Scratch
nag-3-f-ss.bd

File name: nag-3-f-ss.bd

Files of type:

```
C:\Phase-3\Nagawado-Arch\Data-files\Joints\nag
C:\Phase-3\Kasho-Gravity\Data-Files\3-D\Joints\T
C:\Phase-3\Kasho-Gravity\Data-Files\3-D\Joints\F
C:\Phase-3\Kasho-Gravity\Data-Files\3-D\Joints\k
C:\Temp\Kasho-uplift\kas-3-f-ss.bd
C:\Temp\Uplift-test\Uplift-kas.bd
C:\Phase-3\Nagawado-Arch\Data-files\Joints\nag
C:\Phase-3\Nagawado-Arch\Data-files\Joints\Scra
C:\Phase-3\Nagawado-Arch\Data-files\Joints\Scra
C:\Phase-3\Nagawado-Arch\Data-files\Joints\nag
```

Figure 2.2: File Retrieval

Save bd file as Save a bd file for the first time, or with a new name.

Export Will create an `.eps`, `.emf`, `.jpg`, `.bmp` or `.gif` file out of the current main window display.

Switch to Cracker which enables the automatic simulation of crack propagation in 2D only

Chapter 3

Define Boundary

The Define Boundary module enables the user to define all the entities which describe the boundary of the structure to be discritized. Note this section deals only with the geometric definition of the structure, loads, material properties will be specified later.

This module, Fig. 3.1 will generate a `.bd` file which in turn will be executed by the T3D module (developed by Daniel Rypl).

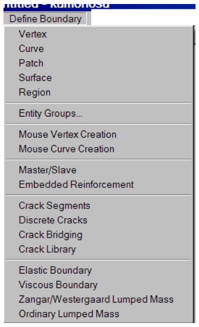


Figure 3.1: Boundary Definition Menu

Vertex to define individual points (called vertices), Sect. 3.1.

Curves are one dimensional curves (or lines) connecting vertices, Sect. 3.2.

Patch Define planar entities composed of 3 or more curves, Sect. 3.3.

Surfaces Define non planar entities composed of 3 or 4 curves, Sect. 3.4.

Regions which are three dimensional objects defined in terms of patches or region, Sect. 3.5.

Entity Groups To lump together various basic entities for easier reference later (such assign a traction to multiple patches), Sect. 3.6.

Mouse Vertex Creation Using a background grid, define vertices with the mouse, Sect. 3.7.

Mouse Curve Creation Using existing vertices, define new curves, Sect. 3.8.

Master/Slave: to tie force entities (vertices, curves, patches or surfaces) to have identical displacements, Sect. 3.11.

Embedded Reinforcement to define steel reinforcement perfectly bonded to the surrounded continuum, Sect. 3.12.

Crack Segments can be line or surface discontinuities, with or without interface elements, Sect. 3.13.1.

Discrete Cracks Define crack entities from previously defined crack segments, Sect. 3.13.2.

Crack Bridging If a reinforcement crosses a crack, Sect. 3.14.

Crack Library (not yet implemented)

Elastic Boundary To apply elastic springs at a vertex, along a curve, or on a patch/surface, Sect. 3.16.

Viscous Boundary To apply a nodal or continuum dashpot at a vertex, along a curve, or on a patch/surface, Sect. 3.17.

Zangaar/Westergaard Lumped Mass To determine and apply added masses along a curve or on a patch/surface, Sect. 3.18.1.

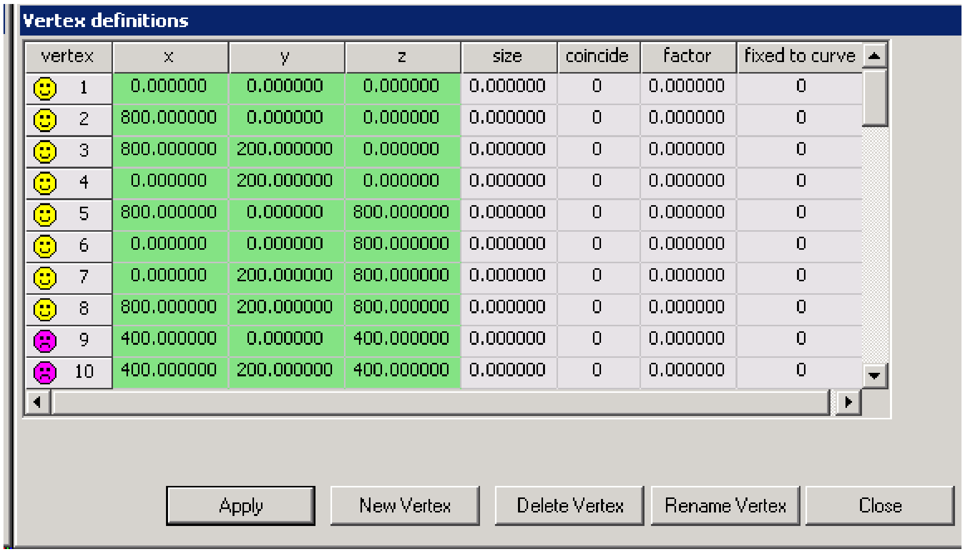
Ordinary Lumped Mass Manually defined, Sect. 3.18.2.

Extrude to 3D To extrude a two dimensional mesh into a three dimensional one, Fig. 3.19.

3.1 Vertex

To define a vertex, the user must specify a vertex number (note vertex numbers need not be sequentially numbered).

To edit an existing vertex, its id must be entered, and then user must click on **Edit**, Fig. 3.2.



	vertex	x	y	z	size	coincide	factor	fixed to curve
☺	1	0.000000	0.000000	0.000000	0.000000	0	0.000000	0
☺	2	800.000000	0.000000	0.000000	0.000000	0	0.000000	0
☺	3	800.000000	200.000000	0.000000	0.000000	0	0.000000	0
☺	4	0.000000	200.000000	0.000000	0.000000	0	0.000000	0
☺	5	800.000000	0.000000	800.000000	0.000000	0	0.000000	0
☺	6	0.000000	0.000000	800.000000	0.000000	0	0.000000	0
☺	7	0.000000	200.000000	800.000000	0.000000	0	0.000000	0
☺	8	800.000000	200.000000	800.000000	0.000000	0	0.000000	0
☹	9	400.000000	0.000000	400.000000	0.000000	0	0.000000	0
☹	10	400.000000	200.000000	400.000000	0.000000	0	0.000000	0

Figure 3.2: Vertex Definition

Mandatory information for vertices:

Vertex id Not necessarily sequential.

Coordinates Vertex coordinates are entered (for a two-dimensional model, the z-coordinate field is left blank) in the grid table.

Optional information:

Size The Size field allows the user to specify a dimension to elements in the vicinity of the vertex. The Size assigned to a vertex takes precedence over the Size assigned during patch definition (discussed later). This value can be left blank/zero.

Coincide Vertex Define the vertex number that is coincident with the current vertex. This option is most often associated with crack definition. Note that if two (or more) vertices have the same coordinates, the vertices with higher numbers are coincident to the vertex with the lowest number. A coincident vertex should have been previously defined.

Factor is a mesh size multiplying factor (default is one, even if the table shows zero) applied to the `-d value` defined as default for the `T3D` mesh generator. Hence, whereas `Size` specifies an absolute size for the mesh (irrespective of the `-d value` defined later), `Factor` is relative to that value.

Fixed to Curve Curve number that the vertex is fixed to. If a vertex is to be located on a curve, but not part of the connectivity of that curve, it is fixed to the curve. It should be noted that by default a vertex fixed to a parent entity inherits mesh size from that entity.

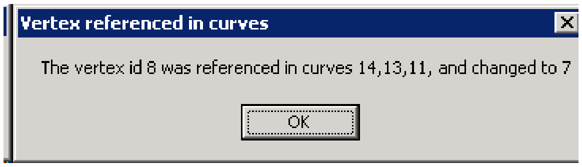
Control keys:

Apply will accept all changes.

New Vertex will generate a new blank row for data entry.

Delete Vertex will delete that entity. Careful, user must check if this vertex is not used in a curve.

Rename a vertex Allows the user to assign a new vertex id. Kumo will inform user if that vertex is used by curves, Fig. 3.3.



The vertex id 8 was referenced in curves 14,13,11, and changed to 7

Figure 3.3: Vertex Rename Warning

Close will simply close the current GUI.

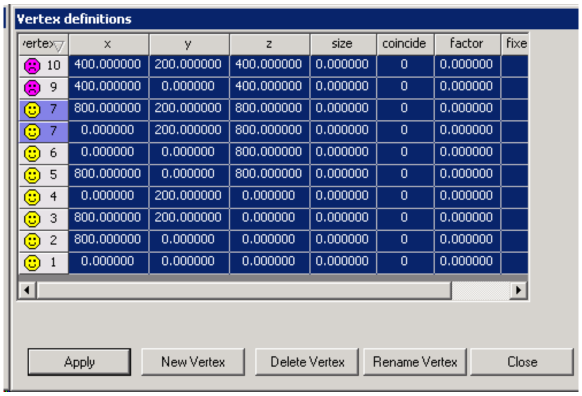
Note:

1. Table can be sorted in ascending/descending order of any of the column values, Fig. 3.4.
2. Dulicate vertex numbers are highlighted.
3. the “smiley” icon means that the vertex is used (at least once) in a curve definition. whereas the “sad” icon means that the vertex is an “orphan” and is not referenced by any curve. (This may happen if the vertex is Fixed on a curve or patch, and is used to have its displacements monitored).
4. User can use the usual Ctrl-C and Ctrl-V to copy and paste into the matrix.

3.2 Curve

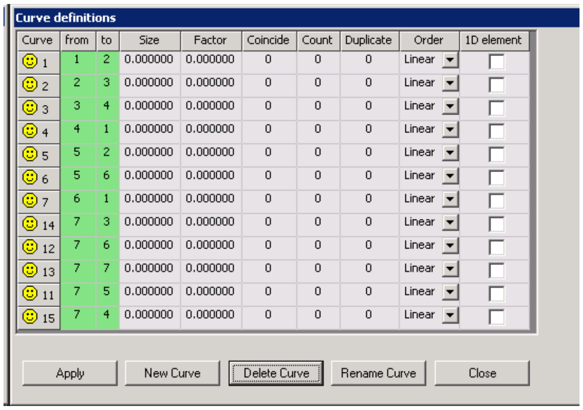
Curve is a compulsory keyword. Mandatory information for all curves, Fig. 3.5: Fig. 3.5.

Curve id : Required for patch, surface, and shell definitions; does not have to be sequential.



ᐯertex	x	y	z	size	coincide	factor	fixe
☹ 10	400.000000	200.000000	400.000000	0.000000	0	0.000000	
☹ 9	400.000000	0.000000	400.000000	0.000000	0	0.000000	
☺ 7	800.000000	200.000000	800.000000	0.000000	0	0.000000	
☺ 7	0.000000	200.000000	800.000000	0.000000	0	0.000000	
☺ 6	0.000000	0.000000	800.000000	0.000000	0	0.000000	
☺ 5	800.000000	0.000000	800.000000	0.000000	0	0.000000	
☺ 4	0.000000	200.000000	0.000000	0.000000	0	0.000000	
☺ 3	800.000000	200.000000	0.000000	0.000000	0	0.000000	
☺ 2	800.000000	0.000000	0.000000	0.000000	0	0.000000	
☺ 1	0.000000	0.000000	0.000000	0.000000	0	0.000000	

Figure 3.4: Vertex Reorder/Duplicate Warning



Curve	from	to	Size	Factor	Coincide	Count	Duplicate	Order	1D element
☺ 1	1	2	0.000000	0.000000	0	0	0	Linear ▼	□
☺ 2	2	3	0.000000	0.000000	0	0	0	Linear ▼	□
☺ 3	3	4	0.000000	0.000000	0	0	0	Linear ▼	□
☺ 4	4	1	0.000000	0.000000	0	0	0	Linear ▼	□
☺ 5	5	2	0.000000	0.000000	0	0	0	Linear ▼	□
☺ 6	5	6	0.000000	0.000000	0	0	0	Linear ▼	□
☺ 7	6	1	0.000000	0.000000	0	0	0	Linear ▼	□
☺ 14	7	3	0.000000	0.000000	0	0	0	Linear ▼	□
☺ 12	7	6	0.000000	0.000000	0	0	0	Linear ▼	□
☺ 13	7	7	0.000000	0.000000	0	0	0	Linear ▼	□
☺ 11	7	5	0.000000	0.000000	0	0	0	Linear ▼	□
☺ 15	7	4	0.000000	0.000000	0	0	0	Linear ▼	□

Figure 3.5: Curve Definition

from-to : Curve start and end vertex numbers (The direction of the curve is from the Vertex i to Vertex j). While the user may assign any direction to the curve, he would be well advised to utilize a convention that facilitates COUNTER-CLOCKWISE element (a patch in two-dimension models) construction.

Optional information

Size The Size field allows the user to specify a dimension to elements in the vicinity of the curve. The Size assigned to a curve takes precedence over the Size assigned during patch definition (discussed later). This value can be left blank/zero.

Factor is a mesh size multiplying factor (default is one, even if the table shows zero) applied to the `-d value` defined as default for the `T3D` mesh generator. Hence, whereas `Size` specifies an absolute size for the mesh (irrespective of the `-d value` defined later), `Factor` is relative to that value.

Coincide Defines a curve number that is coincident with the current curve. This option is most often associated with crack definition. Note that if two (or more) curves have the same coordinates, the curve with higher id is coincident to the curve with the lower id. A coincident curve should have been previously defined.

Count : Define number of quad/hexa elements along the curve (for structured mesh only).

Duplicate : Define that the mesh nodes along this curve must duplicate those of the specified curve.

Curve order : Linear (order 2, default), Quadratic (order 3), Cubic (order 4). Circular, parabolic, hyperbolic and elliptical curves can be defined with curves of order 3 and 4; though the fourth order should used preferably.

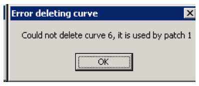
1D element : Specify that this curve is to be included in the mesh as a 1D element. 1D elements (typically steel reinforcement) requires a material ID, specified in the Material ID box.

Control keys:

Apply will accept all changes.

New Curve will generate a new blank row for data entry.

Delete Curve will delete that entity. Prior to deleting, KumoNoSu will check if this curve is not used subsequently, 3.6

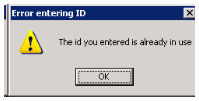


Error deleting curve

Could not delete curve 6, it is used by patch 1

Figure 3.6: Curve Delete Warning

Rename Curve Allows the user to assign a new vertex id. Kumo will inform user if that curve id is already used, Fig. 3.7.



Error entering ID

The id you entered is already in use

Figure 3.7: Curve Rename Warning

Close will simply close the current GUI.

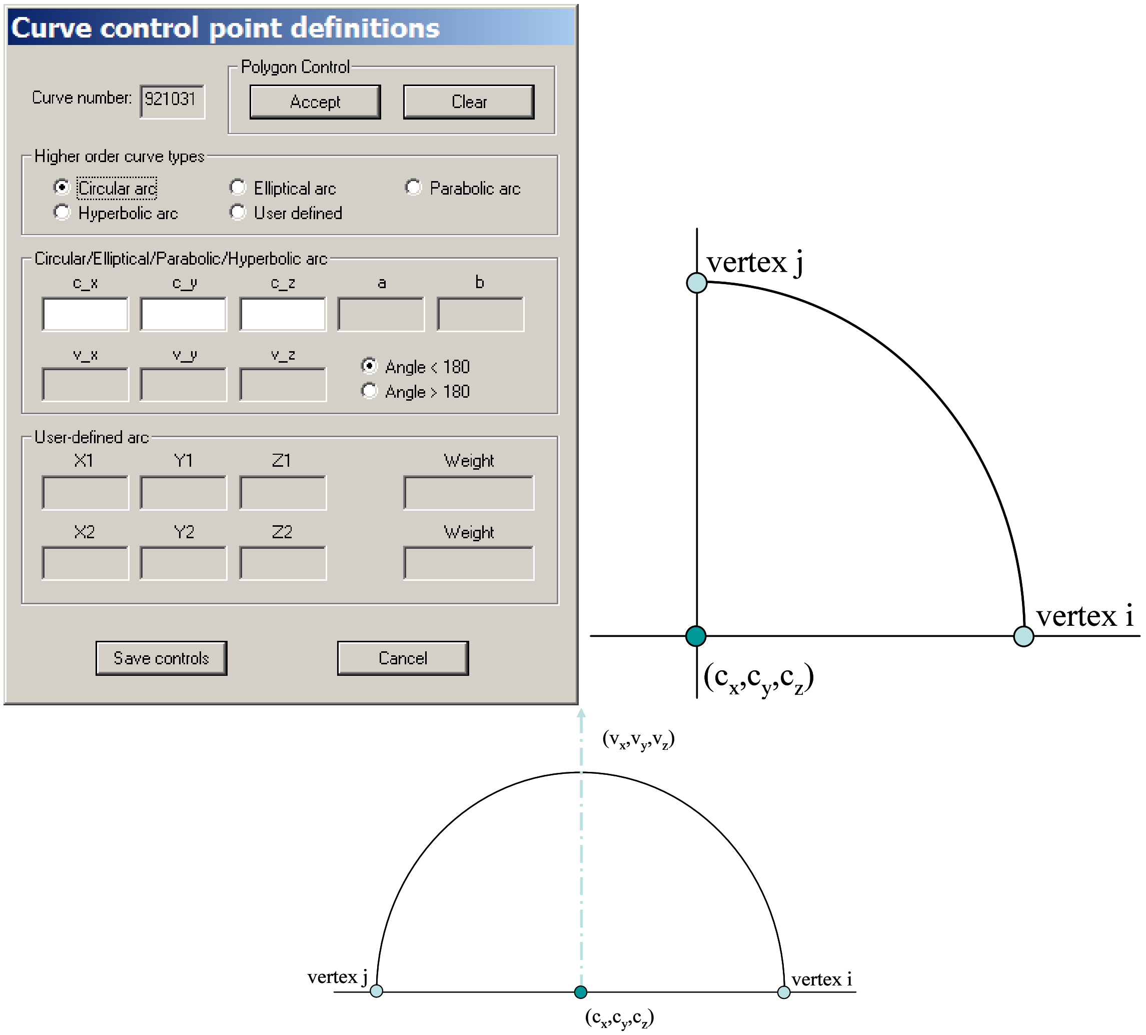
3.2.1 Higher Order Curves

T3D (and thus KumoNoSu) uses rational Bezier curves[1] which are described in details in chapter B

When higher order curves (such as quadratic or cubic) are to be specified a second dialog appears for further definition of control points for the nonlinear curve, KumoNoSu will automatically calculate the control points for (4) types of curves:

Circular arcs : User must specify the center coordinates of the arc, and indicate if the arc is smaller or greater than 180^o. Note that vertices i and j would have been defined in the previous dialog box, Fig. 3.8.

If all three points lie on a straight line, then the arc sustains an angle of 180^o, and then a direction vector $(V_x, V_y, [V_z])$ defines the direction from the circle origin through the center of the arc.



Curve control point definitions

Curve number: 921031

Polygon Control

Higher order curve types: ● Circular arc ○ Elliptical arc ○ Parabolic arc ○ Hyperbolic arc ○ User defined

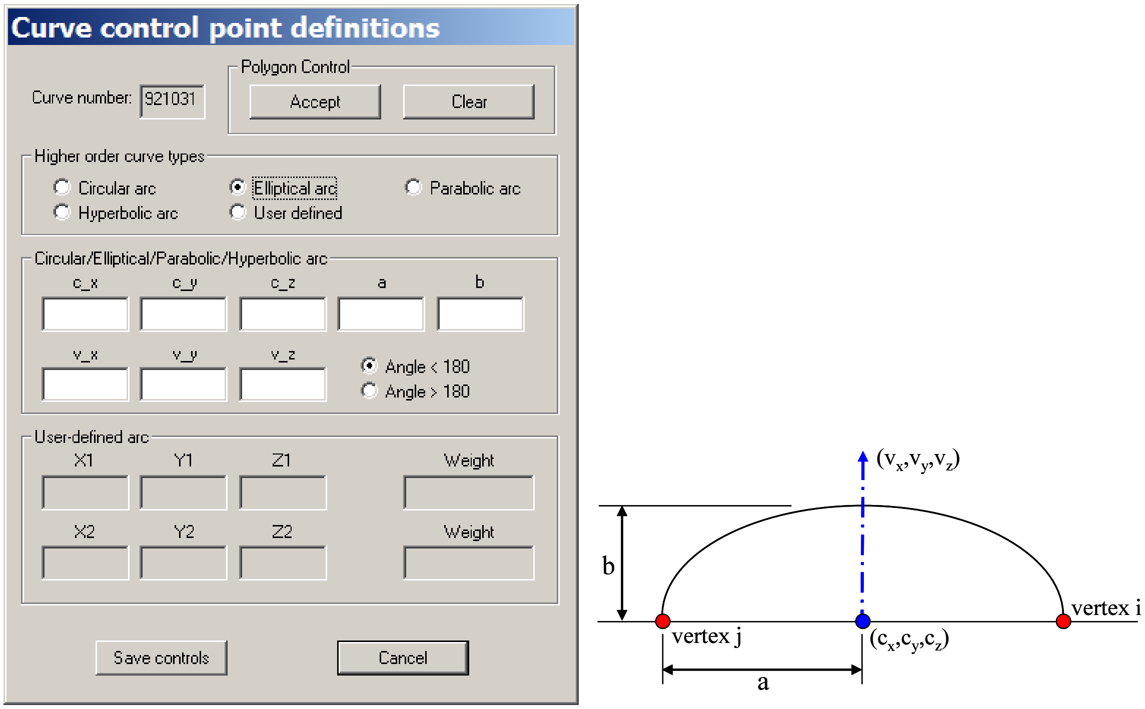
Circular/Elliptical/Parabolic/Hyperbolic arc

● Angle < 180
○ Angle > 180

Figure 3.8: Circular Curve Definition

Elliptical arcs : Defined by the coordinates of the ellipse center $(c_x, c_y, [c_z])$, a and b and the direction vector $(V_x, V_y, [V_z])$ from the center to the center of the arc. Note that vertices i and j would have been defined in the previous dialog box, Fig. 3.9.

[1] Rypl, D., *T3D, Triangularization of 3D Domains*, User Guide, January 2001.



Curve control point definitions

Curve number: 921031

Polygon Control

Higher order curve types

○ Circular arc ● Elliptical arc ○ Parabolic arc

○ Hyperbolic arc ○ User defined

Circular/Elliptical/Parabolic/Hyperbolic arc

c_x c_y c_z a b

v_x v_y v_z

● Angle < 180

○ Angle > 180

User-defined arc

X1 Y1 Z1 Weight

X2 Y2 Z2 Weight

Figure 3.9: Elliptical Curve Definition

Parabolic arcs : The parabola is completely defined by the two vertices i and j (defined in the previous dialogue box), and the parabola vertex $c_x, c_y, [c_z]$), Fig. 3.10.

Hyperbolic arcs : The hyperbola requires definition of a and b as well as a direction vector $(V_x, V_y, [V_z])$. Note that vertices i and j would have been defined in the previous dialog box, Fig. 3.11.

Additionally, control points and weights may be defined by the user.

For subsequent editing of the curve parameters, user should double click on the curve id. Note that if a curve is not linear, the smiley/sad face would have a different color.

3.3 Patch

A patch is a two dimensional section in a plane. A patch can be defined by multiple curves, however all those curves must be co-planar (if not an unforseen error may occur).

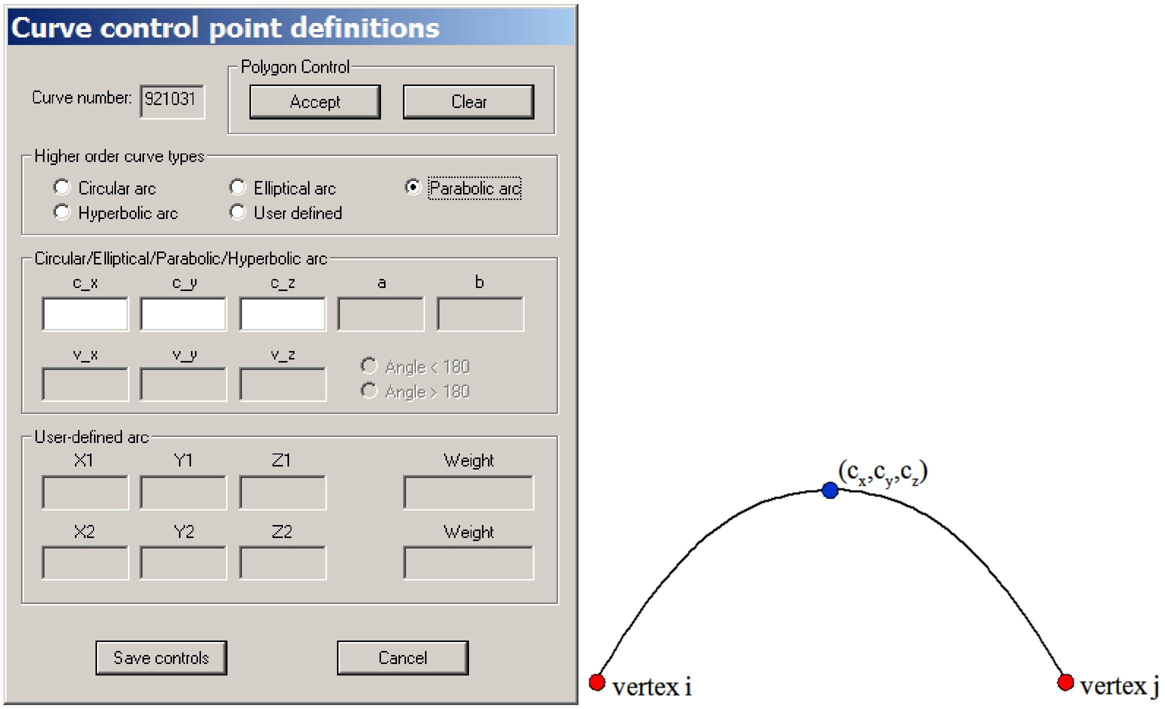
During Patch definition, the direction of the defined members is critical. This direction must be **counterclockwise** with respect to the normal vector (automatically determined by KumoNoSu), and pointing outward.

If the curve definition (from i to j vertex) follows the counter-clockwise orientation, it is considered positive in the boundary curve definition. If the curve definition is oriented clockwise, it is considered negative in the boundary curve definition, Fig. 3.12. Once the data has been accurately added or adjusted for a patch, the user selects the Accept option from the Patch Control sub-menu. Once all vertex data has been entered accurately, the user selects the Save patches button at the bottom of the Patch menu-board.

Mandatory information for all patches, Fig. 3.13: Fig. 3.5:

Patch id number Required for reference in 2D; required for region definitions in 3D; does not have to be sequential.

Boundary curves List of curves completely defining the patch.



Curve control point definitions

Curve number: 921031

Higher order curve types

○ Circular arc ○ Elliptical arc ● Parabolic arc

○ Hyperbolic arc ○ User defined

Circular/Elliptical/Parabolic/Hyperbolic arc

c_x c_y c_z a b

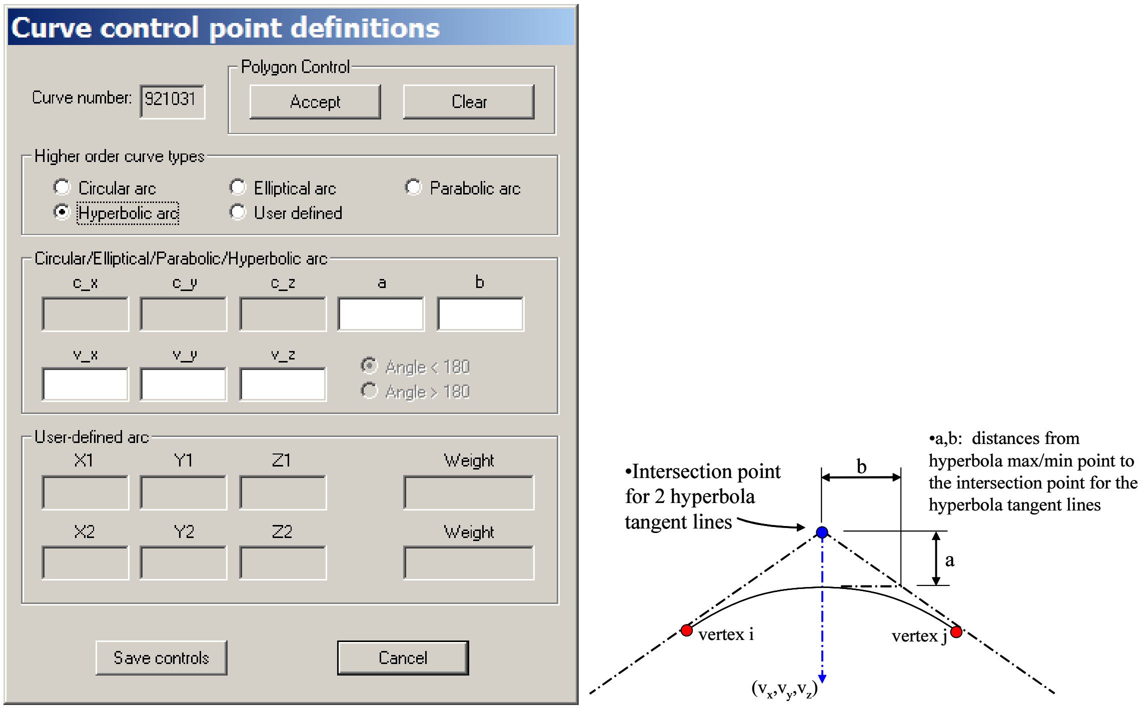
v_x v_y v_z ○ Angle < 180 ○ Angle > 180

User-defined arc

X1 Y1 Z1 Weight

X2 Y2 Z2 Weight

Figure 3.10: Parabolic Curve Definition



Curve control point definitions

Curve number: 921031

Higher order curve types

○ Circular arc ○ Elliptical arc ○ Parabolic arc

● Hyperbolic arc ○ User defined

Circular/Elliptical/Parabolic/Hyperbolic arc

c_x c_y c_z a b

v_x v_y v_z ● Angle < 180 ○ Angle > 180

User-defined arc

X1 Y1 Z1 Weight

X2 Y2 Z2 Weight

Polygon Control
Accept
Clear
Save controls
Cancel
•Intersection point for 2 hyperbola tangent lines
b
•a,b: distances from hyperbola max/min point to the intersection point for the hyperbola tangent lines
a
vertex i
vertex j
(v_x,v_y,v_z)

Figure 3.11: Hyperbola Curve Definition

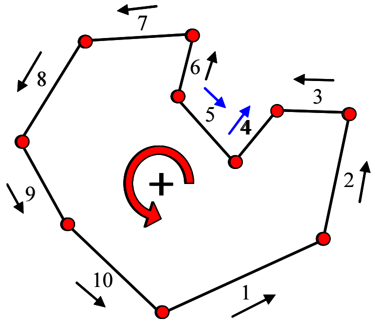
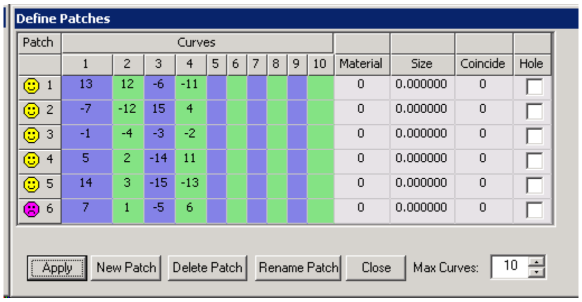


Figure 3.12: Counter-Clockwise Patch Definition



Patch	Curves													
	1	2	3	4	5	6	7	8	9	10	Material	Size	Coincide	Hole
☺ 1	13	12	-6	-11							0	0.000000	0	□
☺ 2	-7	-12	15	4							0	0.000000	0	□
☺ 3	-1	-4	-3	-2							0	0.000000	0	□
☺ 4	5	2	-14	11							0	0.000000	0	□
☺ 5	14	3	-15	-13							0	0.000000	0	□
☹ 6	7	1	-5	6							0	0.000000	0	□

Max Curves: 10

Figure 3.13: Patch Definition

Material Material number associated with the current patch (only for 2D models). Material number must be assigned to two-dimensional patches. The material properties associated with the Material number are defined later from the T3D2Merlin pull-down menu using the Element Groups menu.

Optional selections:

Size Defines mesh size for a patch. The Size field for a patch defines the strong dimension of elements within the vicinity of the patch. The Size assigned within the Patch definition menu is weaker than the previous two Size definitions and always defers to previous definition.

Coincide Defines patch number that is coincident with the current patch. For crack modeling in three-dimensional space, Coincide Patch defines the two surfaces initially occupying the same space.

Hole Defines that this patch is a hole (it will not contain a mesh). This is used for inserting holes in larger patches.

Additional optional parameters may be entered by double clicking on the patch id:

Factor Defines patch mesh size multiplication factor.

Fixed Vertices Defines vertex id of those vertices which lie inside the patch (if any exist).

Fixed Curves Defines curve id of those curves which lie inside the patch (if any exist).

Subpatches Defines the patch numbers of those patches which lie completely inside the patch. For example, hole patches are subpatches (they are inside larger patches).

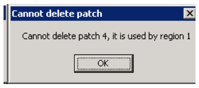
Boundary curves No. 2 Defines a second set of boundary curves (used for special circumstances only).

Control keys:

Apply will accept all changes.

New patch will generate a new blank row for data entry.

Delete patch will delete that entity. Prior to deleting, KumoNoSu will check if this curve is not used subsequently, 3.14



Cannot delete patch 4, it is used by region 1

Figure 3.14: Patch Delete Warning

Rename Curve Allows the user to assign a new patch id. Kumo will inform user if that curve id is already used.

Close will simply close the current GUI.

Max Curves Allows the user to increase the number of allowable curves which define a patch. Careful, if you decrease the current number, you may lose data associated with patch having a large number of curves.

3.4 Surface

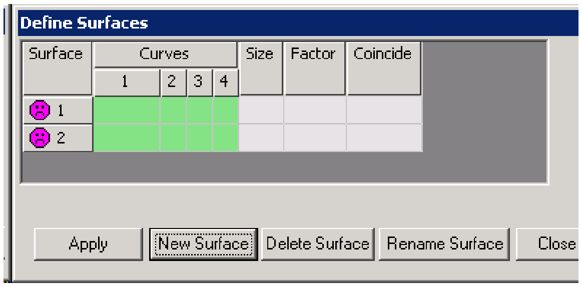
A surface is bound by three or four curves (not necessarily coplanar) defined sequentially (i.e. each curve must be connected to the previously defined one, and the one defined after it).

Curves are specified as positive integers, i.e. we need not worry about continuity of the curves for the surface definition Surface is positive if curves are ordered clockwise around the outer side of the surface.

Curve order in the surface definition matters. Curve direction does not matter.

When defining a region, a surface id may be +ve (if its outward direction is pointing out), or ve (if its outward direction is pointing inside).

Surfaces are only applicable to 3D boundary descriptions, Fig. 3.15.



Surface	Curves				Size	Factor	Coincide
	1	2	3	4			
☹ 1							
☹ 2							

Figure 3.15: Surface Definition

Input data for the Surfaces, are of two types:

Mandatory for all surfaces:

Surface id Required for region definitions in 3D; does not have to be sequential.

Surface curves Three or four curves which define the four sides of the surface. At least one of these curves must be nonlinear, else the surface is planar and is a patch.

Optional

Size Defines mesh size for the surface.

Factor Defines surface mesh size multiplication factor.

Coincide Defines surface number that is coincident with the current surface.

In certain instances, polygon control information will be required for the surface, in which case a second dialog box will open for this information.

It should be noted that the order of the four curves which compose the surface is dependent upon the outer side of the surface (i.e. the side that defines the exterior of the model. Hence, curves are ordered clockwise around the outer side of the surface, and the following rules apply, Fig. 3.16.

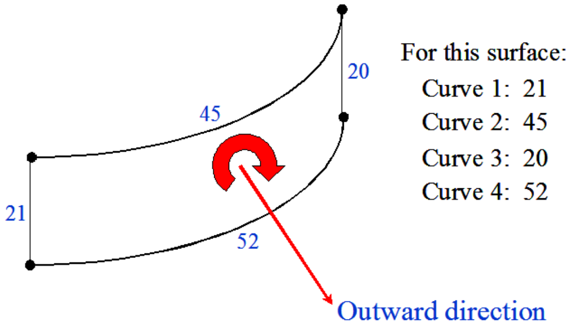


Figure 3.16: Surface Definition

1. If only 1 or 2 **nonconsecutive** curves in a surface are order 3, do not specify a polygon for the surface.

2. If 2 or more **consecutive** curves in a surface are order 3, a polygon must be specified for the surface.

3. If only 1 or 2 **nonconsecutive** curves in a surface are order 4, do not specify a polygon for the surface.

4. If 2 or more **consecutive** curves in a surface are order 4, four polygon coordinates must be specified for the surface.

5. If 1 curve is order 3, and the **previous** or **subsequent** curve is order 4, two polygon coordinates must be specified.

When defining cracks, bounded by two adjacent surfaces then: The two surfaces must be identically defined, i.e the first vertex of the first curve must have the same coordinates for both surfaces, Fig. 3.17.

3.5 Region

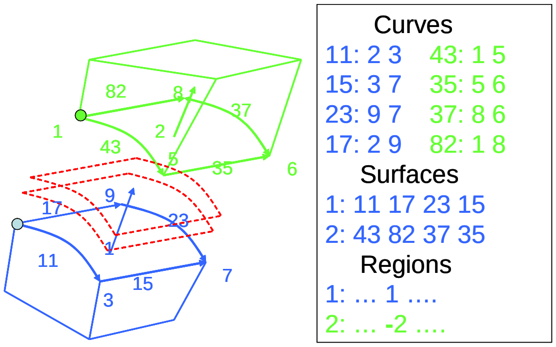
In three-dimensional models the regions must be defined. A region defines a volume. Just as curves comprise boundaries define patches, patches and/or surfaces combine to define the boundaries of a region.

For a region, a patch is defined as positive if its normal points in a positive direction, as defined by the global coordinates. A negative sign during the Boundary patch definition must precede any patch that has a negative unit normal.

The Size defined in the Region definitions is weaker still than the Size defined in the Patch definitions. The user edits regions in a similar fashion as vertices and patches: insert the appropriate number and double click the field box.

Mandatory input parameters for all regions: 3.18.

Region id For reference; does not have to be sequential.



Curves

11: 2 3   43: 1 5

15: 3 7   35: 5 6

23: 9 7   37: 8 6

17: 2 9   82: 1 8

Surfaces

1: 11 17 23 15

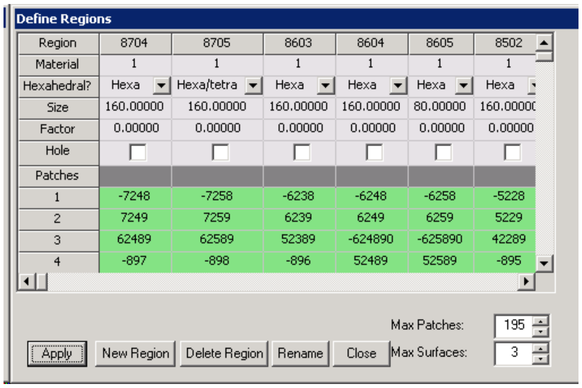
2: 43 82 37 35

Regions

1: ... 1 ....

2: ... -2 ....

Figure 3.17: Surface Crack Definition



Define Regions

Region	8704	8705	8603	8604	8605	8502
Material	1	1	1	1	1	1
Hexahedral?	Hexa	Hexa/tetra	Hexa	Hexa	Hexa	Hexa
Size	160.00000	160.00000	160.00000	160.00000	80.00000	160.00000
Factor	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Hole	□	□	□	□	□	□
Patches						
1	-7248	-7258	-6238	-6248	-6258	-5228
2	7249	7259	6239	6249	6259	5229
3	62489	62589	52389	-624890	-625890	42289
4	-897	-898	-896	52489	52589	-895

Max Patches: 195

Max Surfaces: 3

Figure 3.18: Surface Definition

Material Material number associated with the current region.

Hexahedral User may specify if the elements to be generated in the current regions are Tetrahedrons (default), Hexahedral, or a combination of the two.

Boundary patches List of patches composing the boundary of the region (if any exist.

Boundary surfaces List of surfaces composing the boundary of the region (if any exist).

Optional entries are

Size Define mesh size for the region.

Factor Define region mesh size multiplication factor.

Fixed curves Define any curves which lie inside the region and are not connected to any patch or surface.

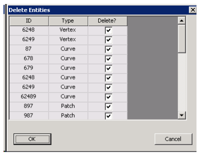
Hole Indicating that we have a three-dimensional cavity.

Control keys:

Apply will accept all changes.

New Region will generate a new blank row for data entry.

Delete Region will delete that entity. Prior to deleting, KumoNoSu will provide the user with a list of all entities pertaining to this region, 3.19. The user can then delete all those entitities, or uncheck those which should be retained.



Delete Entities

ID	Type	Delete?
6248	Vertex	☑
6249	Vertex	☑
87	Curve	☑
678	Curve	☑
679	Curve	☑
6248	Curve	☑
6249	Curve	☑
62489	Curve	☑
897	Patch	☑
987	Patch	☑

Figure 3.19: Region Delete Warning

Rename Allows the user to assign a new patch id. Kumo will inform user if that curve id is already used.

Close will simply close the current GUI.

Max Patches/Surfaces Allows the user to increase the number of allowable patches/surfaces which define a patch. Careful, if you decrease the current number, you may lose data associated with patch having a large number of curves.

Certain important rules apply for region definition, Fig. 3.20:

1. Boundary surface numbers are always positive.
2. Boundary patch numbers are positive if the patchs normal points OUT of the region.
3. Boundary patch numbers are negative if the patchs normal points INTO the region.
4. Since the surface normal (defined by the order of the surface curves) is always out of the region, boundary surface numbers are always positive.

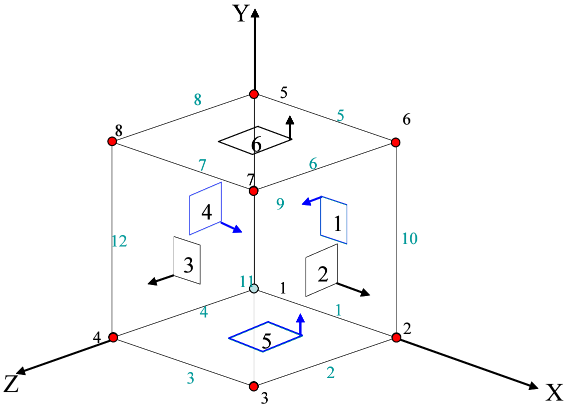


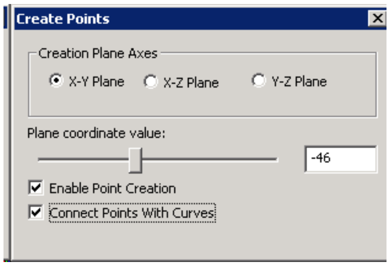
Figure 3.20: Region Definition

3.6 Entity Groups

NEED TO FIX KUMO Entity groups enable the user to lump or group together various entities whcih will subsequently inherit the same characteristic (particularly in the load definition).

3.7 Mouse Vertex Creation

User can use the mouse to define new vertices and have them connected by curves, Fig. 3.21. However, user must first select a projection plane, and then specify the third coordinate through the slider. At that point, when the left button of the mouse is pressed a new vertex is created at the closest grid point. Grid resolution must be defined within the View Option in Section 4.1.5. Note that the mouse current position is echoed in the bottom toolbar. If **Enable Point Creation** is active, then pressing the mouse creates the new vertex. If **Connect Points with Curves** is active, then sequential curves are defined.


Create Points

Creation Plane Axes

● X-Y Plane ○ X-Z Plane ○ Y-Z Plane

Plane coordinate value: -46

☑ Enable Point Creation

☑ Connect Points With Curves

Figure 3.21: Vertex Definition by the Mouse

3.8 Mouse Curve Creation

This option, Fig. 3.22 enables the user to select individual vertices with the mouse and have them connected by curves.

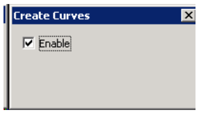


Figure 3.22: Curve Definition by the Mouse

3.9 Curve Selection

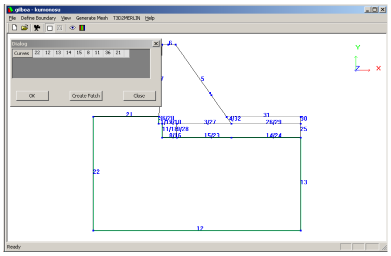
This option, Fig. 3.23 enables the user to select curves by clicking the curve number with the left button of the mouse. Once selected, the curve is assigned a new color. The list of curves is first saved in a buffer inside the dialogue box.

It is important to note that the first curve selected should be oriented in a counterclockwise direction for the patch to be selected (i.e. it will be defined as a positive integer in the patch definition).

To deactivate a selection, user must click again on a previously selected curve.

When the user selects `Create` a new patch is created with the selected curves. Kumo will attempt to determine the sign of the curve id's (based on the first one defined) such that a counterclockwise patch is defined.

User should open the patch definition menu, and assign to the newly created patch the proper parameters (such as material group if 2D analysis).



Curves	22	12	13	14	15	8	11	36	21

Figure 3.23: Curve Selection with the Mouse

3.10 Patch/Surface Selection

This option, Fig. 3.24 enables the user to select patches by clicking the patch number with the left button of the mouse. The list of patches is first saved in a buffer inside the dialogue box.

A powerful feature, is to automatically add all the patches connected to the previously selected one. User has simply to keep on clicking on `Add Face` and kumo will identify the adjacent patches connected to the previously defined one. This process can be repeated recursively.

It is important to note that the first patch selected should have an outward normal.

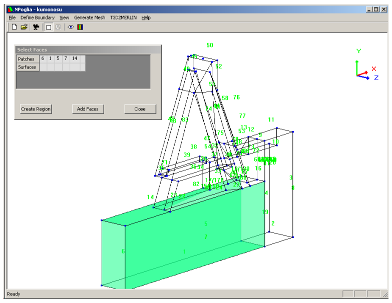
To deactivate a selection, user must click again on a previously selected patch.

When the user selects `Create` a new region is created with the selected patches/surfaces. Kumo will attempt to determine the sign of the patches/surfaces id's (based on the first one defined) such that the outward normal ones have a positive id number.

User should open the region definition menu, and assign to the newly created region the proper parameters (such as material group).

3.11 Master/Slave

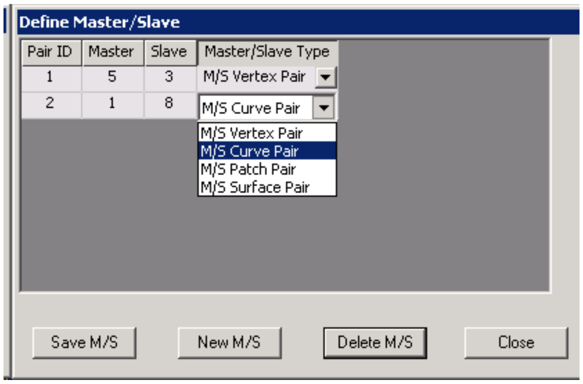
Master/slave nodes in the FE mesh may be defined in KumoNoSu through the master/slave pair definition dialog. Both vertex pairs and curve pairs may be defined in the dialog. In the case of M/S curve



Patches	6	1	5	7	14
Surfaces					

Figure 3.24: Patch/Surface Selection with the Mouse

pairs, all nodes in the FE mesh that coincide with this curve will be M/S nodes, Fig. 3.25. Mandatory information:



Pair ID	Master	Slave	Master/Slave Type
1	5	3	M/S Vertex Pair
2	1	8	M/S Curve Pair

Figure 3.25: Master-Slave Definition

Master/Slave counter This id will never be explicitly referenced, it is just a counter

Master/Slave pairs.

Entity type Specify if the M/S pair is a vertex, curve, patch, or surface pair (patch and surface pairs in 3D only).

NOTE: all vertices or curves defined as M/S must also be defined as **coinciding** entities. Fig. 3.26 is an example of master slave definition.

3.12 Embedded Reinforcement

Embedded reinforcement, to be added to the MERLIN input file, may be defined in KumoNoSu through the reinforcement definitions dialog, Fig. 3.27.

Two methods are available for embedded reinforcement: definition by existing vertices or definition by end coordinates. Either of these options may be selected in the 'Definition type' section. If definition by vertices is chosen, the information is imputed in the 'Rebar vertices information section'. The material number for the reinforcement, beginning vertex number (Vert$_i$i), and ending vertex number (Vert$_j$) must be defined. If definition by coordinates (x, y, z) is selected, the information is inputted in the 'Rebar (x, y, z) information' section. The material number, (x, y, z) coordinates for the beginning (i) vertex, and (x, y, z) coordinates for the end (j) vertex must be defined. After the requisite information is imputed, click 'Accept' to save the reinforcement definition. When all rebars have been defined, click 'Save reinf' to save the information and close the dialog box.

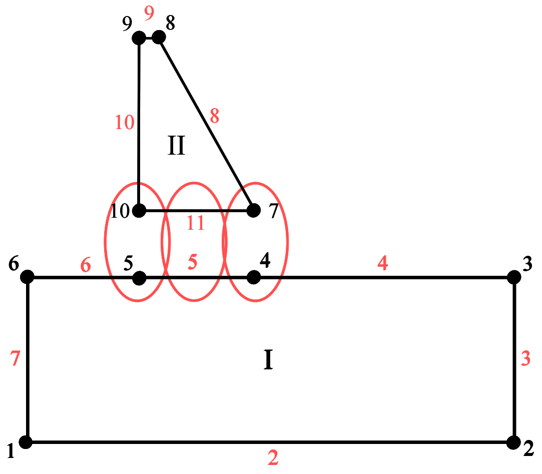
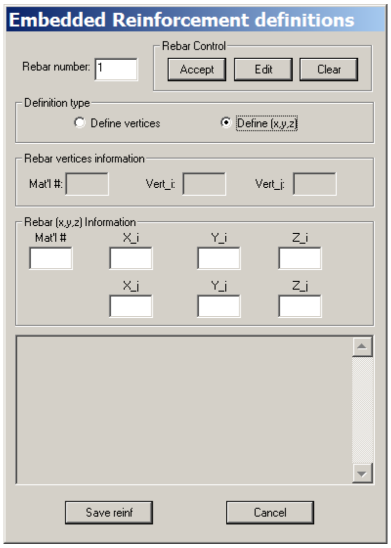


Figure 3.26: Master-Slave Definition Example



Embedded Reinforcement definitions

Rebar number: 1

Rebar Control

Accept | Edit | Clear

Definition type

○ Define vertices ● Define (x,y,z)

Rebar vertices information

Mat'l #: Vert_i: Vert_j:

Rebar (x,y,z) Information

Mat'l # X_i Y_i Z_i

X_j Y_j Z_j

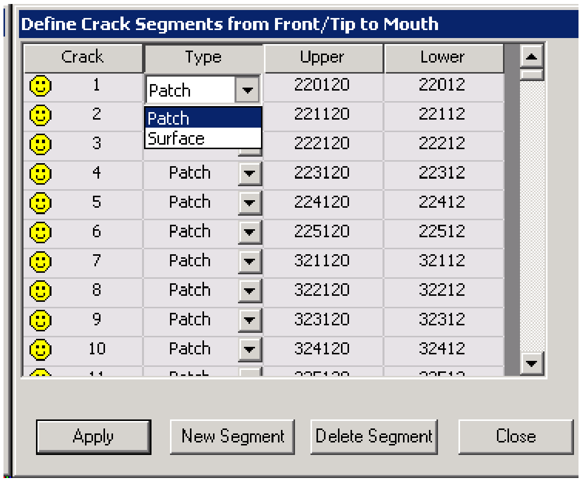
Save reinf | Cancel

Figure 3.27: Embedded Reinforcement

3.13 Cracks

3.13.1 Crack Segments

A Crack is composed of one or multiple segments. The first step consists in defining those pairs of segments which will later constitute a crack, Fig. 3.28:



Crack	Type	Upper	Lower
1	Patch	220120	22012
2		221120	22112
3		222120	22212
4	Patch	223120	22312
5	Patch	224120	22412
6	Patch	225120	22512
7	Patch	321120	32112
8	Patch	322120	32212
9	Patch	323120	32312
10	Patch	324120	32412
11	Patch	325120	32512

Figure 3.28: Crack Definition

Type which can be a curve, patch or surface.

Upper/Lower entities.

Note:

1. Crack paths are always defined from crack front to crack mouth.
2. Curve orientation (i and j vertices) in each segment must be from crack front to crack mouth.
3. The two curves or patches composing each crack segment must be defined as **Coincide**.
4. 2D Crack segment upper and lower curves are prescribed according to the location of the crack front and Fig. 3.29.

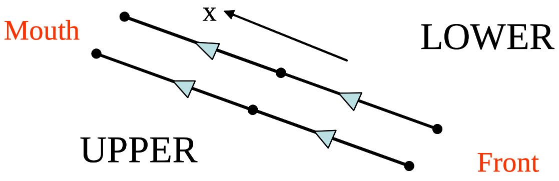


Figure 3.29: Crack Orientation Definition for 2D Cases

5. 3D Crack segment upper and lower patches are prescribed according to which patch is on top when the crack opens (in terms of the positive global coordinate directions), Fig. 3.30.

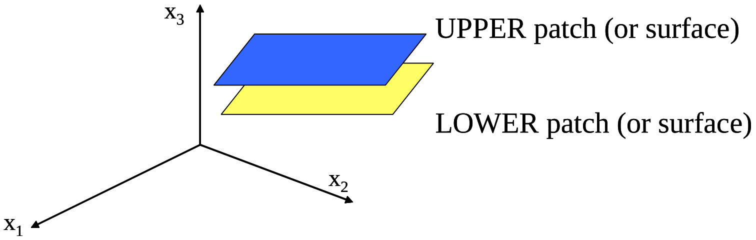
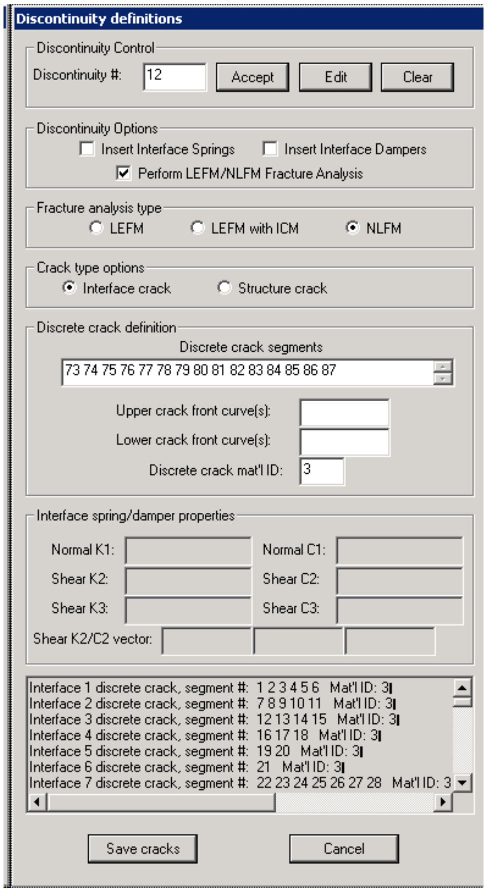


Figure 3.30: Crack Orientation Definition for 3D cases



Discontinuity Control

Discontinuity #: 12

Discontinuity Options

☐ Insert Interface Springs ☐ Insert Interface Dampers

☑ Perform LEFM/NLFM Fracture Analysis

Fracture analysis type

○ LEFM ○ LEFM with ICM ● NLFM

Crack type options

● Interface crack ○ Structure crack

Discrete crack definition

Discrete crack segments: 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87

Upper crack front curve(s):

Lower crack front curve(s):

Discrete crack mat'l ID: 3

Interface spring/damper properties

Normal K1: Normal C1:

Shear K2: Shear C2:

Shear K3: Shear C3:

Shear K2/C2 vector:

```
Interface 1 discrete crack, segment #:  1 2 3 4 5 6   Mat'l ID: 3
Interface 2 discrete crack, segment #:  7 8 9 10 11   Mat'l ID: 3
Interface 3 discrete crack, segment #:  12 13 14 15   Mat'l ID: 3
Interface 4 discrete crack, segment #:  16 17 18   Mat'l ID: 3
Interface 5 discrete crack, segment #:  19 20   Mat'l ID: 3
Interface 6 discrete crack, segment #:  21   Mat'l ID: 3
Interface 7 discrete crack, segment #:  22 23 24 25 26 27 28   Mat'l ID: 3
```

Figure 3.31: Discrete Crack Definition

3.13.2 Discrete Cracks

One the crack segments have been defined, the User may now define the discrete cracks, Fig. 3.31.

Discontinuity id is the id associated with the crack/discontinuity about to be defined.

Discontinuity Option Partially implemented

Insert Interface Spring between the two lips of the discontinuity. Not yet implemented.

Insert Interface Damper between the two lips of the discontinuity. Not yet implemented.

Perform LEFM/NLFM Analysis, is by default the option.

Fracture Analysis Type Linear Elastic Fracture Mechanics (LEFM), Nonlinear Fracture Mechanics (NLFM), or LEFM with interface crack elements (cohesive crack model).

Crack Type Option Interface cracks are between two patches in two-dimensional analyses and between two regions in three-dimensional analyses. A structure crack develops within a patch or a region.

Discrete Crack Definition

Discrete crack segments lists the crack segments id's constituting the discrete crack.

Upper crack front curve Vertex (or curve in 3D) at the upper surface crack front.

Lower crack front curve Vertex (or curve in 3D) at the lower surface crack front.

Discrete Crack Mat'l ID to be used only if interface elements are to be inserted along the crack (NLFM or LEFM with ICM options).

Interface spring/damper properties Inactive.

Note that identification of the upper and lower curves is for the proper application of the uplift (FERC model).

3.13.3 Examples

Fig. 3.32 is an example of 2D structure crack where the Upper crack surface is curve 6, the lower one is 7, the crack front is 7. The crack is defined by only one crack segment.

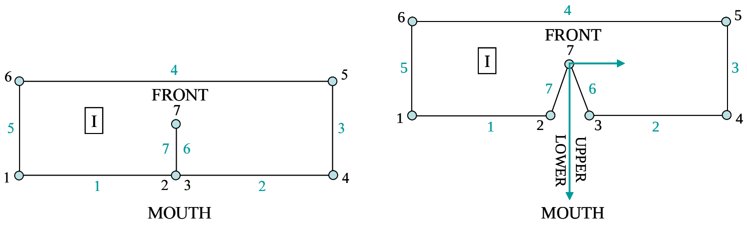


Figure 3.32: 2D Example of a Structure Crack

Fig. 3.33 is an example of 3D structure crack. The crack segment is composed of the pair of patches I and II, the crack front is defined by curve 5. The upper segment is II (because it is "above" patch I in the positive X direction), the lower one is I.

A 2D interface crack is shown in Fig. 3.34. Vertex 11, 12, 13 must coincide with vertex 1, 2, 3 respectively (Vertex 11, 12, 13 must be defined AFTER vertex 1, 2,3). Curves direction MUST be defined from tip (or front) to the mouth of the crack Curves 14 and 15 must coincide with curves 4 and 5 respectively. (Curves 14 and 15 must be defined AFTER curves 4 and 5.

Another example of 2D interface crack is shown in Fig. 3.35.

Fig. 3.36 is an example of 3D interface crack.

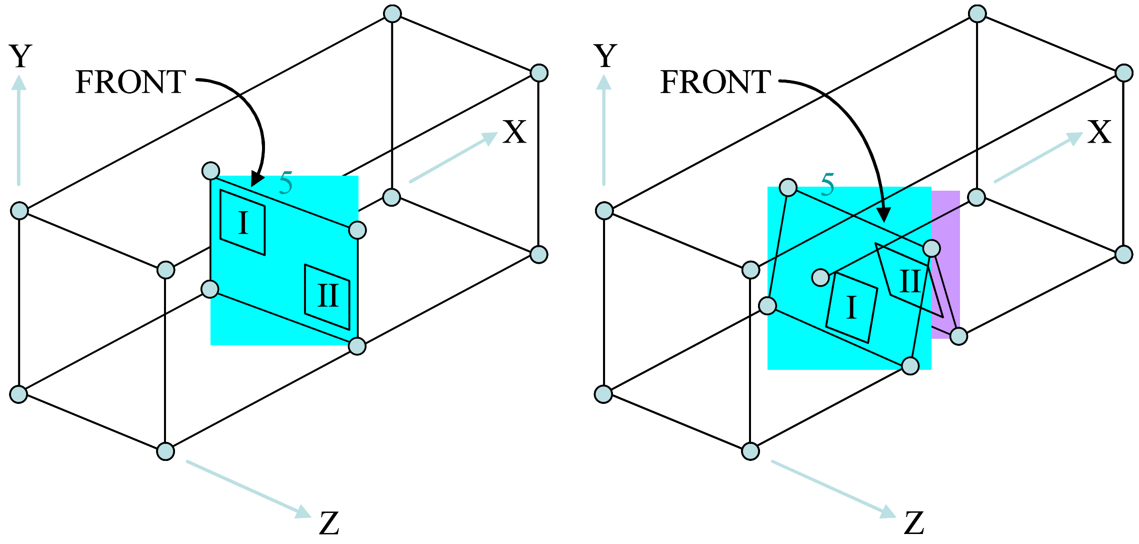


Figure 3.33: 3D Example of a Structure Crack

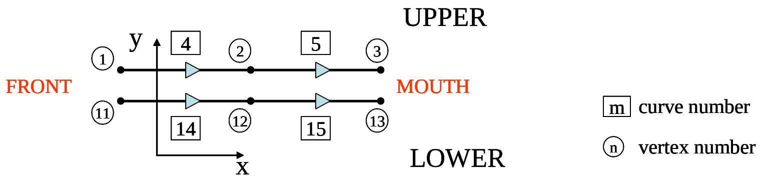


Figure 3.34: 2D Example of an Interface Crack, Upper and Lower

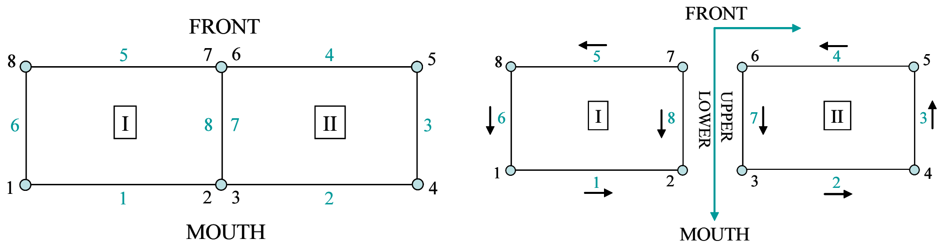


Figure 3.35: 2D Example of an Interface Crack, Upper and Lower

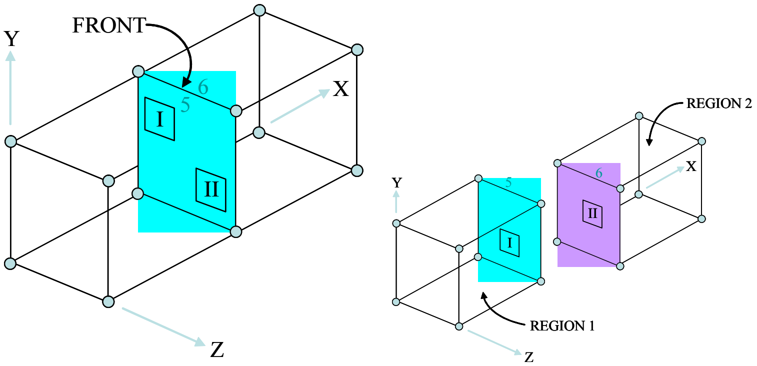
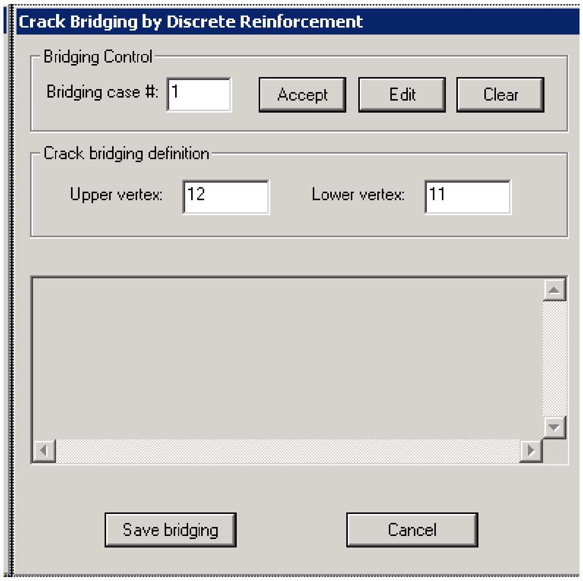


Figure 3.36: 3D Example of an Interface Crack

3.14 Crack Bridging a Truss Element

In reinforced concrete analysis, a crack may cross a truss element modeling a rebar, Fig. 3.37.


Crack Bridging by Discrete Reinforcement

Bridging Control

Bridging case #: 1

Crack bridging definition

Upper vertex: 12

Lower vertex: 11

Figure 3.37: Rebar Crossing a Crack

Case No. For reference; does not have to be sequential.

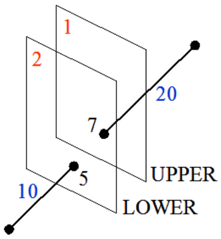
Upper vertex Vertex number of the curve end which lies on the upper surface of the crack.

Lower vertex Vertex number of the curve end which lies on the lower surface of the crack

Fig. 3.38 is a simple illustrative example.

3.15 Crack Library

Inactive



If curves 10 and 20 are intended to bridge the crack formed by patches 1 and 2, then vertex 7 is the upper vertex and vertex 5 is the lower vertex for the bridging reinforcement definition

Figure 3.38: Example of Rebar Crossing a Crack

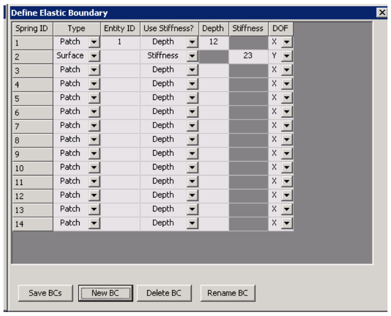
3.16 Elastic Boundary

Elastic Boundary enables the user to specify elastic springs along a global direction (X, Y or Z) on a vertex, along a curve or over a patch.

The spring stiffness can be determined either from

$$K = \frac{AE}{t} \tag{3.1}$$

where A is the tributary area of the spring (automatically determined by KumoNoSu) on the basis of adjacent elements, E is the Young's modulus of the adjoining material (automatically detected by KumoNoSu), and t is the effective *thickness* to be specified by the user, or can be explicitly given, Fig. 3.39.



Spring ID	Type	Entity ID	Use Stiffness?	Depth	Stiffness	DOF
1	Patch	1	Depth	12		X
2	Surface		Stiffness		23	Y
3	Patch		Depth			X
4	Patch		Depth			X
5	Patch		Depth			X
6	Patch		Depth			X
7	Patch		Depth			X
8	Patch		Depth			X
9	Patch		Depth			X
10	Patch		Depth			X
11	Patch		Depth			X
12	Patch		Depth			X
13	Patch		Depth			X
14	Patch		Depth			X

Figure 3.39: Elastic Boundary

User is reminded that the spring connects an entity to a rigid support.

3.17 Viscous Boundary

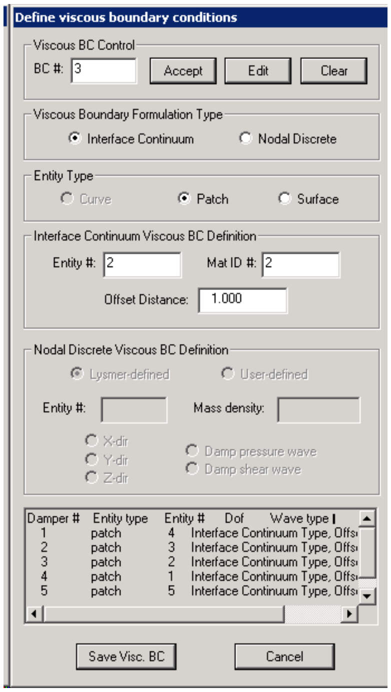
3.17.1 Discrete Dashpots/Nodal

3.17.2 Continuous Dashpots/Elements

Viscous boundaries absorbs the energy from pressure and shear waves, (**?**). Damping of these waves is based upon the elastic properties of the continuum along the boundary

$$C = \rho V_i A \left\{ \begin{array}{rll} \text{Shear Wave:} & V_s & = \sqrt{\frac{G}{\rho}} \\ \text{Pressure Wave:} & V_p & = \frac{1}{s} V_s ; s^2 = \frac{1-2\nu}{2(1-\nu)} \end{array} \right. \tag{3.2}$$

Mandatory information for all viscous boundaries, Fig.3.40



Define viscous boundary conditions

Viscous BC Control

BC #: 3

Viscous Boundary Formulation Type

● Interface Continuum ○ Nodal Discrete

Entity Type

○ Curve ● Patch ○ Surface

Interface Continuum Viscous BC Definition

Entity #: 2 Mat ID #: 2

Offset Distance: 1.000

Nodal Discrete Viscous BC Definition

● Lysmer-defined ○ User-defined

Entity #: Mass density:

○ X-dir
○ Y-dir
○ Z-dir

○ Damp pressure wave
○ Damp shear wave

Damper #	Entity type	Entity #	Dof	Wave type
1	patch	4	Interface Continuum Type, Offs	
2	patch	3	Interface Continuum Type, Offs	
3	patch	2	Interface Continuum Type, Offs	
4	patch	1	Interface Continuum Type, Offs	
5	patch	5	Interface Continuum Type, Offs	

Figure 3.40: Viscous Boundary

B.C id boundary condition number, for reference.

Formulation Interface continuum will generate continuum dashpots elements along the selected entity. In 2D these will be four noded elements, and in 3D 6 or 8 noded elements.

Entity type (curve in 2D, patch or surface in 3D).

Interface Continuum parameters

Entity No. to identify the curve, patch or surface.

Mat ID Material group associated with the viscous interface elements (to be defined later).

Offset Distance The interface dashpot elements have a zero thickness formulation. Hence, user can select an arbitrary offset distance for visualization.

Nodal Discrete parameters

Entity No. number of the entity at which to apply viscous B.C.s.

Mass density mass density γ of the material on the viscous boundary (if zero or blank, the γ of the actual material will be used).

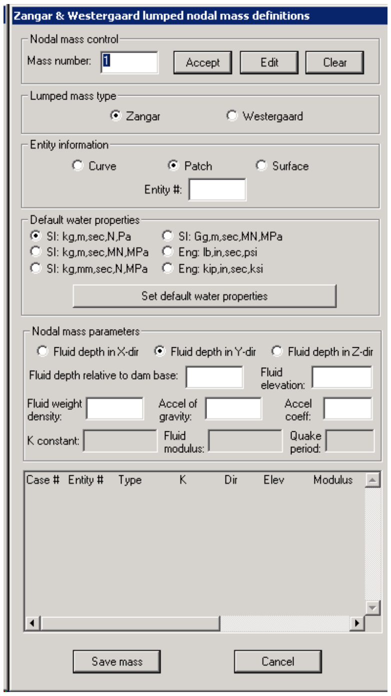
X-dir, Y-dir, [Z-dir] degree of freedom to be damped.

Damp pressure wave/Damp shear wave wave to be damped.

3.18 Lumped Masses

3.18.1 Westergaard-Zangaar

Mandatory information for all Westergaard (vertical upstream face) or Zangaar (inclined upstream face) added mass, Fig.3.41



Mass number: 1

● Zangar ○ Westergaard

○ Curve ● Patch ○ Surface

Entity #:

● SI: kg,m,sec,N,Pa ○ SI: Gg,m,sec,MN,MPa
○ SI: kg,m,sec,MN,MPa ○ Eng: lb,in,sec,psi
○ SI: kg,mm,sec,N,MPa ○ Eng: kip,in,sec,ksi

○ Fluid depth in X-dir ● Fluid depth in Y-dir ○ Fluid depth in Z-dir

Fluid depth relative to dam base:
Fluid elevation:
Fluid weight density:
Accel of gravity:
Accel coeff:
K constant:
Fluid modulus:
Quake period:

Case #	Entity #	Type	K	Dir	Elev	Modulus

Figure 3.41: Westergaard Added Mass Definition

Nodal mass number for reference.

Lumped Mass type Westergaard or Zangar.

Entity type Curve (in 2D) or Patch (in 3D).

Entity No. entity number.

Axis defining reservoir depth X, Y (or Z in 3D).

K **constant** K constant for Westergaard equation (defined by Westergaard as 51.0 lb/ft^3 or 8011.4 N/m^3).

Water elevation Elevation of the reservoir surface (note that this is not the relative depth of the reservoir, but rather the elevation of the surface).

Water modulus Elastic modulus of the water (may be taken as 300 kips/in^2 or 2.068 GPa).

Fluid weight weight per unit volume of the fluid (Westergaard only).

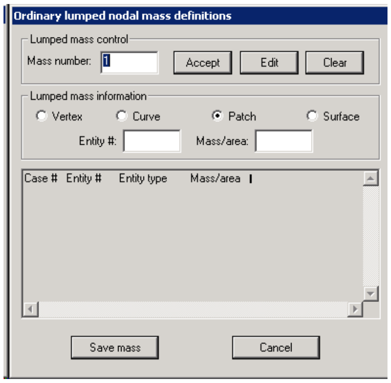
Accel of gravity Acceleration of gravity.

Quake period Period of the earthquake motion (Westergaard only).

Relative fluid depth Relative depth of the reservoir.

3.18.2 User-Defined

This feature must be checked.



Mass number: 1

○ Vertex ○ Curve ● Patch ○ Surface

Figure 3.42: Lumped Mass Definition

3.19 Extrude

The extrude capability, Fig. 3.43 enables the user to begin with a 2D mesh definition, and then extrude it into a 3D mesh along the z axis by a user specified length.

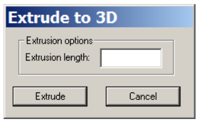


Figure 3.43: Extrude Interface

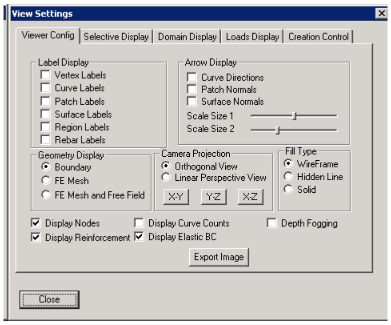
Chapter 4

View

4.1 View Settings

4.1.1 Viewer Config

Fig. 4.1...



Label Display

- ☐ Vertex Labels
- ☐ Curve Labels
- ☐ Patch Labels
- ☐ Surface Labels
- ☐ Region Labels
- ☐ Rebar Labels

Arrow Display

- ☐ Curve Directions
- ☐ Patch Normals
- ☐ Surface Normals
- Scale Size 1
- Scale Size 2

Geometry Display

- ● Boundary
- ○ FE Mesh
- ○ FE Mesh and Free Field

Camera Projection

- ● Orthogonal View
- ○ Linear Perspective View

Fill Type

- ● WireFrame
- ○ Hidden Line
- ○ Solid

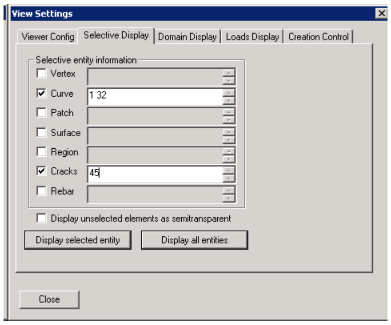
☑ Display Nodes ☐ Display Curve Counts ☐ Depth Fogging

☑ Display Reinforcement ☑ Display Elastic BC

Figure 4.1: Viewer Configuration

4.1.2 Selective Display

Fig. 4.2 ....



Selective entity information

- ☐ Vertex
- ☑ Curve 1 32
- ☐ Patch
- ☐ Surface
- ☐ Region
- ☑ Cracks 45
- ☐ Rebar

☐ Display unselected elements as semitransparent

Figure 4.2: Selective Display

4.1.3 Domain Display

Fig. 4.3 ...

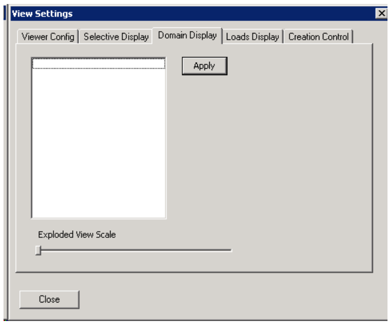
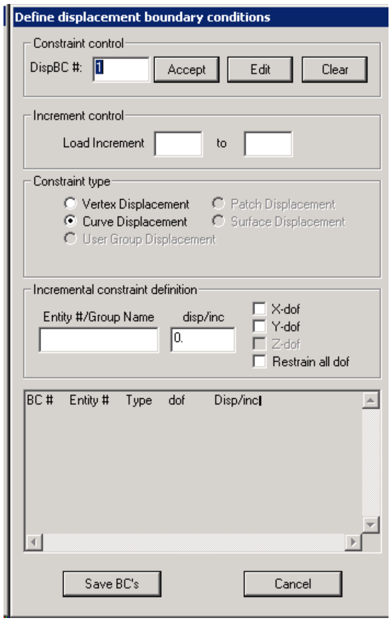


Figure 4.3: Domain Display

4.1.4 Load Display

Fig. 4.4 ....


Define displacement boundary conditions

Constraint control

DispBC #: 1

Increment control

Load Increment ___ to ___

Constraint type

○ Vertex Displacement ○ Patch Displacement
● Curve Displacement ○ Surface Displacement
○ User Group Displacement

Incremental constraint definition

Entity #/Group Name ___ disp/inc 0.

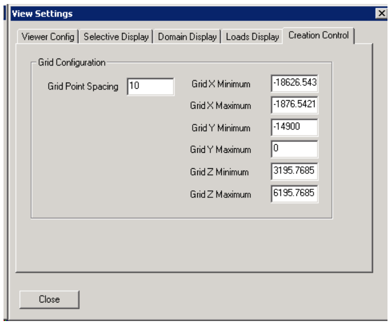
□ X-dof
□ Y-dof
□ Z-dof
□ Restrain all dof

BC # Entity # Type dof Disp/inc

Figure 4.4: Load Display

4.1.5 Creation Control

Fig. 4.5 ....



Grid Configuration

Grid Point Spacing: 10

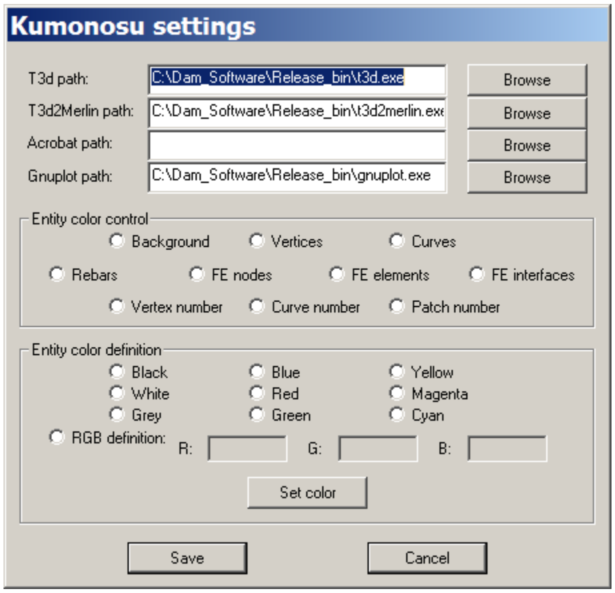
Grid X Minimum: -18626.543
Grid X Maximum: -1876.5421
Grid Y Minimum: -14900
Grid Y Maximum: 0
Grid Z Minimum: 3195.7685
Grid Z Maximum: 6195.7685

Figure 4.5: Creation Control

4.2 Kumonosu Settings

4.3 Settings

The user may wish to set or adjust the Preprocessor settings, Fig. 4.6. This enables the user to specify:



T3d path: C:\Dam_Software\Release_bin\t3d.exe
T3d2Merlin path: C:\Dam_Software\Release_bin\t3d2merlin.exe
Acrobat path:
Gnuplot path: C:\Dam_Software\Release_bin\gnuplot.exe

Entity color control
○ Background ○ Vertices ○ Curves
○ Rebars ○ FE nodes ○ FE elements ○ FE interfaces
○ Vertex number ○ Curve number ○ Patch number

Entity color definition
○ Black ○ Blue ○ Yellow
○ White ○ Red ○ Magenta
○ Grey ○ Green ○ Cyan
○ RGB definition: R: G: B:

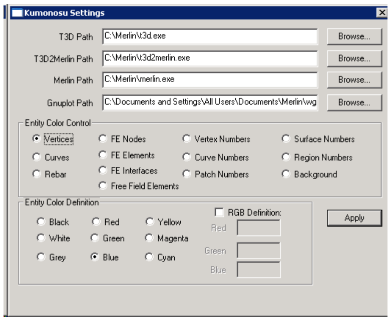
Figure 4.6: KumoNoSu Setting

1. Paths where the supporting software resides (T3d, T3d2Merlin, Acrobat Reader and Gnuplot).
2. Colors of various entities used by KumoNoSu .

4.4 Lighting

Fig. 4.8 ...

4.5 Reset Camera



Kumonosu Settings

T3D Path: C:\Merlin\t3d.exe

T3D2Merlin Path: C:\Merlin\t3d2merlin.exe

Merlin Path: C:\Merlin\merlin.exe

Gnuplot Path: C:\Documents and Settings\All Users\Documents\Merlin\wg

Entity Color Control

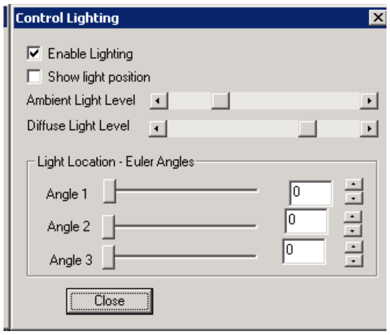
● Vertices ○ FE Nodes ○ Vertex Numbers ○ Surface Numbers
○ Curves ○ FE Elements ○ Curve Numbers ○ Region Numbers
○ Rebar ○ FE Interfaces ○ Patch Numbers ○ Background
○ Free Field Elements

Entity Color Definition

○ Black ○ Red ○ Yellow
○ White ○ Green ○ Magenta
○ Grey ● Blue ○ Cyan

□ RGB Definition:
Red
Green
Blue

Figure 4.7: Kumonosu Settings



Control Lighting

☑ Enable Lighting

□ Show light position

Ambient Light Level

Diffuse Light Level

Light Location - Euler Angles

Angle 1: 0

Angle 2: 0

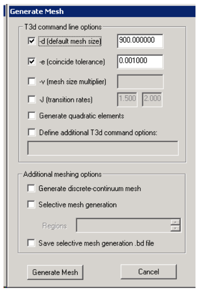
Angle 3: 0

Figure 4.8: Light Settings

Chapter 5

Generate Mesh

sectionGenerate Mesh The Generate Mesh dialog box, Fig. 5.1 offers the user the opportunity to modify the default mesh size and the coincide tolerance.



Generate Mesh

T3d command line options

☑ -d (default mesh size) 900.000000

☑ -e (coincide tolerance) 0.001000

□ -v (mesh size multiplier)

□ -J (transition rates) 1.500 2.000

□ Generate quadratic elements

□ Define additional T3d command options:

Additional meshing options

□ Generate discrete-continuum mesh

□ Selective mesh generation

Regions

□ Save selective mesh generation .bd file

Figure 5.1: Mesh Generation

d indicates to the code a mesh size for nodes, curves, patches and regions that have not had their respective sizes explicitly set.

e represents the allowable tolerance for coincident nodes. Two nodes within this specified tolerance will be treated as coincident nodes.

v check

-J check for tetra/hexa?

Quadratic elements buggy

Additional T3D commands can be specified. Please consult the T3D manual.

Discrete-Continuum will place an interface element in between all the edges of the elements.

Selective Mesh Generation enables the user to test the mesh by meshing only selected regions.

Save selective mesh `.bd` file for testing/debugging purposes.

Chapter 6

T3D2MERLIN

After successful generation of the mesh by T3d (`.t3d` file), the user must first define a control file (`.ctrl`) and then run T3d2Merlin to create a Merlin input file. All of the information required by the control file may be defined from the T3d2Merlin pull-down menu. The menu, Fig. 6.1 lists:

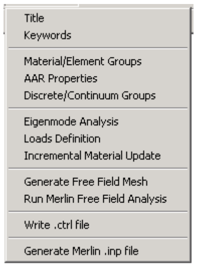


Figure 6.1: T3D2Merlin Menu

Title Sect. 6.1.

Keyword to specify the control commands, Sect. 6.2.

Material/Element Groups For material definition, Sect. 6.3.

AAR Properties Sect. 6.4.

Discrete/Continuum Groups To enable the generation mesh with interface elements in between all the elements, Sect. 6.5.

Eigenmode Analysis Sect. 6.6.

Loads Definition Sect. 6.7.

Incremental Material Update to modify material properties within load increments, Sect. 6.8.

Generate Free Field for dynamic analysis with radiation damping and active free field, Sect. 6.9.

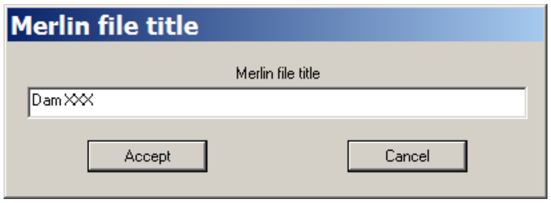
Run Free Field Perform the finite element analysis of the free fields, and define the boundary conditions for the mesh, Sect. 6.10.

Write .ctrl File Save the cntrol file, Sect. 6.11.

Generate Merlin .inp file , Sect. 6.12

6.1 Title

The user may provide a description the contents of the input file within this field. This line will be written verbatim into the beginning of the Merlin input file, Fig. 6.2.



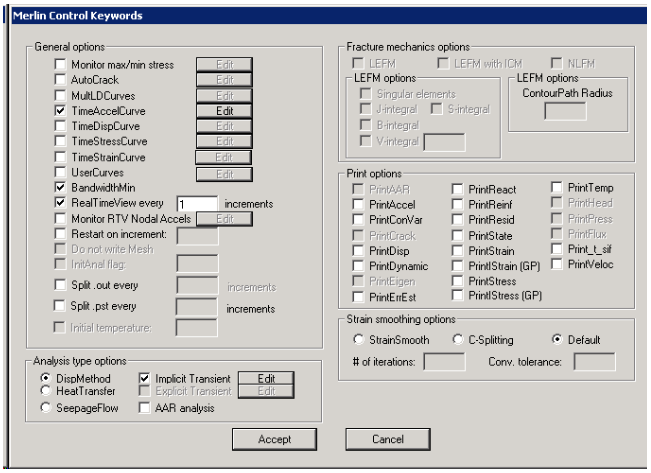
Merlin file title: Dam XXX

Figure 6.2: Title Data Entry

It is strongly recommended to include the selected units in the Title.

6.2 Keywords

The Keywords-menu allows the user to define general control options for the subsequent Merlin analysis, Fig. 6.3.



General options
- □ Monitor max/min stress
- □ AutoCrack
- □ MultLDCurves
- ☑ TimeAccelCurve
- □ TimeDispCurve
- □ TimeStressCurve
- □ TimeStrainCurve
- □ UserCurves
- ☑ BandwidthMin
- ☑ RealTimeView every 1 increments
- □ Monitor RTV Nodal Accels
- □ Restart on increment:
- □ Do not write Mesh
- □ InitAnal flag:
- □ Split .out every increments
- □ Split .pst every increments
- □ Initial temperature:

Analysis type options
- ● DispMethod
- ○ HeatTransfer
- ○ SeepageFlow
- ☑ Implicit Transient
- □ Explicit Transient
- □ AAR analysis

Fracture mechanics options
- □ LEFM
- □ LEFM with ICM
- □ NLFM

LEFM options
- □ Singular elements
- □ J-integral
- □ S-integral
- □ B-integral
- □ V-integral

LEFM options
- ContourPath Radius

Print options
- □ PrintAAR
- □ PrintAccel
- □ PrintConVar
- □ PrintCrack
- □ PrintDisp
- □ PrintDynamic
- □ PrintEigen
- □ PrintErrEst
- □ PrintReact
- □ PrintReinf
- □ PrintResid
- □ PrintState
- □ PrintStrain
- □ PrintIStrain (GP)
- □ PrintStress
- □ PrintIStress (GP)
- □ PrintTemp
- □ PrintHead
- □ PrintPress
- □ PrintFlux
- □ Print_t_sif
- □ PrintVeloc

Strain smoothing options
- ○ StrainSmooth
- ○ C-Splitting
- ● Default
- # of iterations:
- Conv. tolerance:

Figure 6.3: Keywords Specification

6.2.1 General options

6.2.1.1 Monitor Max/Min Stress

Specify which maximum stress component is to be monitored throughout the analysis and its maximum reported (and corresponding location) reported for each increment.

Merlin: ??

6.2.1.2 AutoCrack

This keyword is associated with the module Cracker which drives Merlin for automatic crack propagation. When Merlin senses that either a crack is to nucleate, or a crack must extend, thus requiring remeshing to accomodate discrete cracks, then control is passed back from Merlin to Cracker, Fig. 6.5.

Merlin: 1.1.12

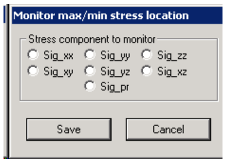

Monitor max/min stress location
Stress component to monitor
○ Sig_xx ○ Sig_yy ○ Sig_zz
○ Sig_xy ○ Sig_yz ○ Sig_xz
○ Sig_pr
Save
Cancel

Figure 6.4: Monitor Maximum Stress

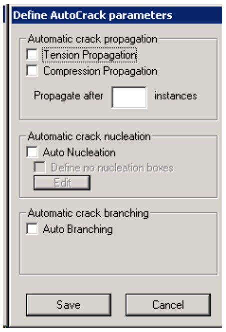

Define AutoCrack parameters
Automatic crack propagation
□ Tension Propagation
□ Compression Propagation
Propagate after
instances
Automatic crack nucleation
□ Auto Nucleation
□ Define no nucleation boxes
Edit
Automatic crack branching
□ Auto Branching
Save
Cancel

Figure 6.5: Auto-Crack Specification

Automatic Propagation Allows merlin to remesh if a crack wants to propagate. The user can specify the number of instances that crack stability requirements have been violated before remeshing takes place. Violation occurs if the stress intensity factor exceeds the fracture toughness in LEFM, or if the crack tip tensile stress exceeds the tensile strength of the material in NLFM.

Automatic Crack Nucleation Allows Merlin to interrupt the analysis if at some points the maximum tensile principal stress exceeds the tensile strength. If desired, the user can specify boxes inside which no crack nucleation can take place.

Automatic Crack Branching Will allow crack bifurcation.

6.2.1.3 `MultLDCurves`

Merlin: 1.1.11.2

This option will allow the user to track the load displacement response of a set of nodes, Fig. 6.6.

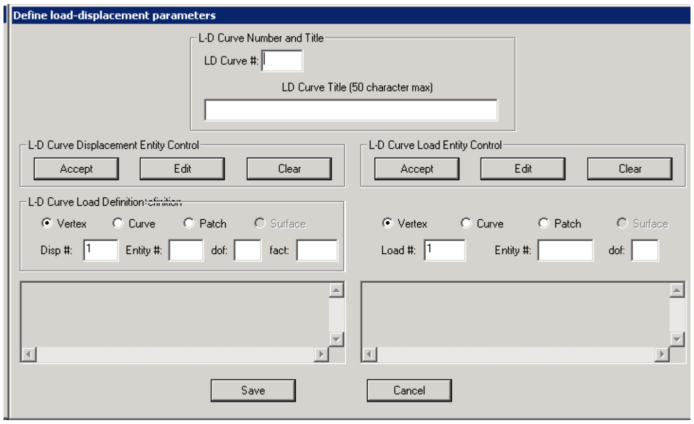
As data entry for this option can be confusing, the user should understand that:

1. One or more curves can be specified.
2. Each curve will allow the user to specify load and displacement.
3. The displacement can be either one of a specific node/dof, or the algebraic sum of multiple node/dof pairs. This is important if one needs a crack opening displacement (then one facor would be +1, and the other -1), or the displacement of a point with respect to the base. This corresponds to the Disp. #.
4. The load can be either the load at one vertex, or the entire load applied along a curve or over a patch. Again, at time one may need the resultant of a traction load.

Disp # or Load # Number of the current displacement or load monitoring point (for reference).

Entity type type of entity at which to monitor loads or displacements (vertex, curve, or patch).

LD Curve # Curve number that the current displacement or load monitoring point belongs to.



Define load-displacement parameters

L-D Curve Number and Title

LD Curve #:

LD Curve Title (50 character max)

L-D Curve Displacement Entity Control

L-D Curve Load Definition·efinition

● Vertex ○ Curve ○ Patch ○ Surface

Disp #: 1 Entity #: dof: fact:

L-D Curve Load Entity Control

● Vertex ○ Curve ○ Patch ○ Surface

Load #: 1 Entity #: dof:

Figure 6.6: Load Displacement Curve Definition

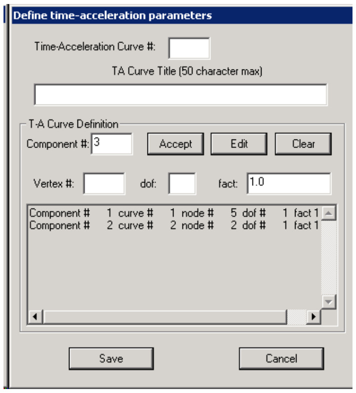
Entity # number of the current entity.

dof degree of freedom to monitor (1-2-3).

fact Factor to be applied to the displacement (note, factors are only defined for displacements).

6.2.1.4 `TimeAccelCurve`

A single time-acceleration curve may monitor the accelerations vs. time (or increments) at one or more vertices, 6.7:



Define time-acceleration parameters

Time-Acceleration Curve #:

TA Curve Title (50 character max)

T-A Curve Definition

Component #: 3

Vertex #: dof: fact: 1.0

```
Component #   1  curve #   1  node #   5  dof #   1  fact 1
Component #   2  curve #   2  node #   2  dof #   1  fact 1
```

Figure 6.7: Time Acceleration Curve Definition

Case # Number of the current monitoring point (for reference).

Curve Title to identify it amongst other plots.

TA Curve # Curve number that the current monitoring point belongs to.

Vertex # Vertex number of the current monitoring point.

dof Degree of freedom to monitor (acceleration).

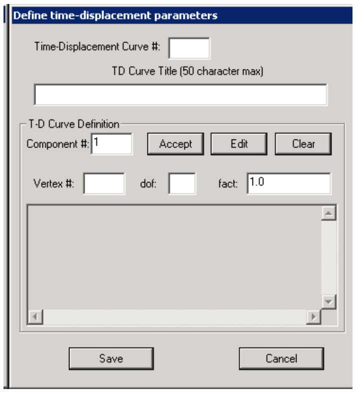
compoment Stress component to monitor.

fact Factor to be applied to the acceleration/displacement/stress at the current monitoring point.

6.2.1.5 TimeDispCurve

Merlin: 1.1.11.3

Definition of one or more time-displacement is performed using the Define time-displacement parameters dialogs, 6.8:


Define time-displacement parameters
Time-Displacement Curve #:
TD Curve Title (50 character max)
T-D Curve Definition
Accept
Edit
Clear
Vertex #:
dof:
Save
Cancel

Component #: 1

fact: 1.0

Figure 6.8: Time Displacement Curve Definition

Case # Number of the current monitoring point (for reference).

Title to be assigned to the curve.

TA Curve # Curve number that the current monitoring point belongs to.

Vertex # Vertex number of the current monitoring point.

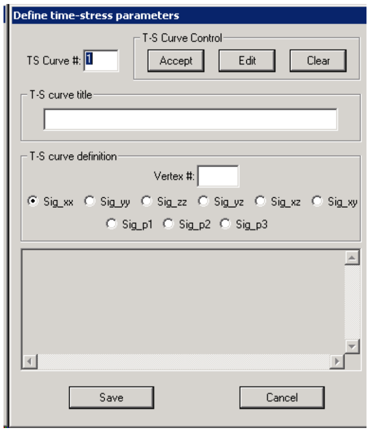
dof Degree of freedom to monitor.

fact Factor to be applied to the displacement at the current monitoring point.

6.2.1.6 Time Stress Curve

Merlin: 1.1.11.6

Definition of one or more time-stress is performed using the Define time-stress parameters dialogs, 6.9, (`TimeStrsCrv` option in Merlin):


Define time-stress parameters
T-S Curve Control
Accept
Edit
Clear
T-S curve title
T-S curve definition
Vertex #:
○ Sig_yy ○ Sig_zz ○ Sig_yz ○ Sig_xz ○ Sig_xy
○ Sig_p1 ○ Sig_p2 ○ Sig_p3
Save
Cancel

TS Curve #: 1

● Sig_xx

Figure 6.9: Time Stress Curve Definition

Case # Number of the current monitoring point (for reference).

TA Curve # Curve number that the current monitoring point belongs to.

Title to be assigned to the curve.

Vertex # Vertex number of the current monitoring point.

component Stress component to monitor.

fact Factor to be applied to the acceleration/displacement/stress at the current monitoring point (may be useful to convert units).

6.2.1.7 `Time Strain Curve`

Definition of one or more time-strain is performed using the Define time-strain parameters dialogs, 6.10, (`TimeStrnCrv` option in Merlin):

Figure 6.10: Time Strain Curve Definition

Case # Number of the current monitoring point (for reference).

TA Curve # Curve number that the current monitoring point belongs to.

Title to be assigned to the curve.

Vertex # Vertex number of the current monitoring point.

component Stress component to monitor.

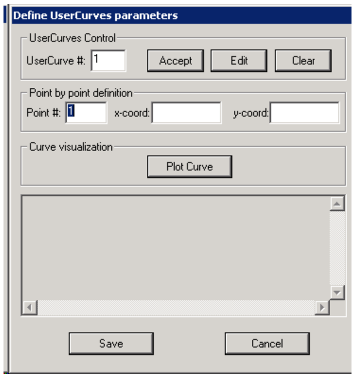
fact Factor to be applied to the acceleration/displacement/strain at the current monitoring point (may be useful to convert units).

6.2.1.8 `UserCurves`

Merlin: 1.1.7

In various instances of Merlin, User may define specific x-y curves (such as for G_F, Dynamic uplift etc). This option enables the user to explicitly specify one or more curve, which will then be referenced in the input data.

6.11



Define UserCurves parameters

UserCurve #: 1

Point #: 1

Figure 6.11: User Defined Curve

6.2.1.9 `BandwidthMin`

This is an optional setting that allows the user to minimize the time Merlin spends decomposing the global stiffness matrix.

6.2.1.10 `RealTimeView`

Merlin: 1.1.13

Requests the generation of an external file containing the displacements and accelerations at selected nodes every n increments. Merlin will then generate an `.rtv` file for real time view during the analysis by Spider.

6.2.1.11 `Restart on Increment`

Merlin: 1.1.3

This option enables Merlin to use the previously generated `.pst` file to restart at a certain increment. This is most useful when first a static analysis is performed, followed by a dynamic one.

6.2.1.12 `Do Not Write Mesh`

This option specifies that for the current Merlin input file, there is no need to write the mesh data, as those may be coming from the restart file.

6.2.1.13 `InitAnalflag`

Missing

6.2.1.14 Split Output

Merlin: 1.7.21

In some analysis, the ASCII output file may be too large for the operating system or for the editor. This option enables the user to split the output into multiple files, each containing the results of n increments.

6.2.1.15 Split .pst

In some analysis, the .pst output file may be too large for the operating system. This option enables the user to split the .pst file into multiple files, each containing the results of n increments.

6.2.1.16 Initial Temperature

Merlin: 1.1.10

This is the stress free initial temperature, needed only for AAR analysis.

6.2.2 Analysis Type Options

DispMethod tells the code that the analysis will be traditional, structural, finite element analysis.

Heat Transfer allows the user to specify that a heat transfer analysis be to be performed. Selection of this option will preclude selection of analysis options relating to stress or seepage problems.

Merlin: 1.3.6

SeepageFlow alerts the computer to an impending seepage flow analysis problem.

Thermo-elast option not yet available.

Poro-elast option not yet available.

6.2.2.1 `Implicit Transient`

Merlin: 1.3.4

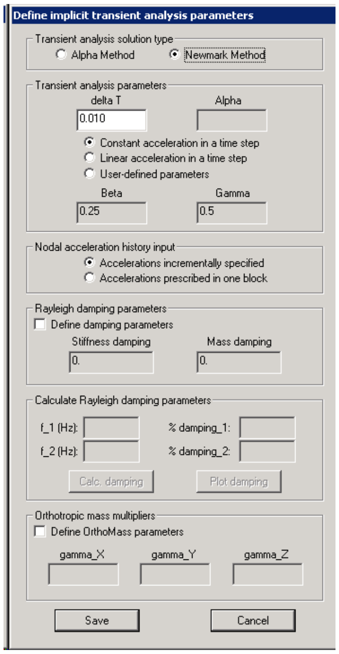
For transient analysis, user must specify, Fig. 6.12:

Integration Scheme Newmark's β method or Hughes α method.

delta T corresponding to the ΔT of the transient analysis.

Hughes alpha coefficient.

Accelerations models during the time step (to automatically determine Newmark's coefficients).



Define implicit transient analysis parameters

Transient analysis solution type

○ Alpha Method ● Newmark Method

Transient analysis parameters

delta T: 0.010

Alpha:

● Constant acceleration in a time step

○ Linear acceleration in a time step

○ User-defined parameters

Beta: 0.25

Gamma: 0.5

Nodal acceleration history input

● Accelerations incrementally specified

○ Accelerations prescribed in one block

Rayleigh damping parameters

□ Define damping parameters

Stiffness damping: 0.

Mass damping: 0.

Calculate Rayleigh damping parameters

f_1 (Hz): % damping_1:

f_2 (Hz): % damping_2:

Calc. damping Plot damping

Orthotropic mass multipliers

□ Define OrthoMass parameters

gamma_X gamma_Y gamma_Z

Save Cancel

Figure 6.12: Data Entry for Transient Analysis

Newmark beta β coefficient for Newmark's time integration (only if user defined). Note that KumoNoSu predefines β as 1/4 and 1/6 for constant and linear acceleration over the time step.

Newmark gamma γ coefficient for Newmark's time integration (only if user defined). Note that KumoNoSu predefines β as 1/2 and 1/2 for constant and linear acceleration over the time step.

Nodal Acceleration Definition check if accelerations are defined in each increment or if the accelerations are prescribed in a single block.

Stiffness damping stiffness damping parameter α for Rayleigh damping

Mass damping mass damping parameter β for Rayleigh damping

Compute Rayleigh Damping coefficients. to facilitate determination of the Rayleigh damping coefficients, user can simply specify the two frequencies f [Hz] and corresponding damping [%], KumoNoSu will then compute the coefficients, and plot them Fig. 6.13.

OrthoMass Parameters if an earthquake is not aligned with a principal axis, user can use **orthomass** to define an orthotropic mass matrix. the two (or three) γ coefficients, each less than 1., premultiply the mass matrix along that direction.

6.2.2.2 `Explicit Transient`

Merlin: 1.3.5

Initial time step time increment which should be very carefully determined to ensure stability.

Coefficient for critical timestep see Merlin Manual.

Stiffness and Mass Damping for Rayleigh damping.

Nodal Acceleration Definition check if accelerations are defined in each increment or if the accelerations are prescribed in a single block.

Merlin: 3.4.1.10.2

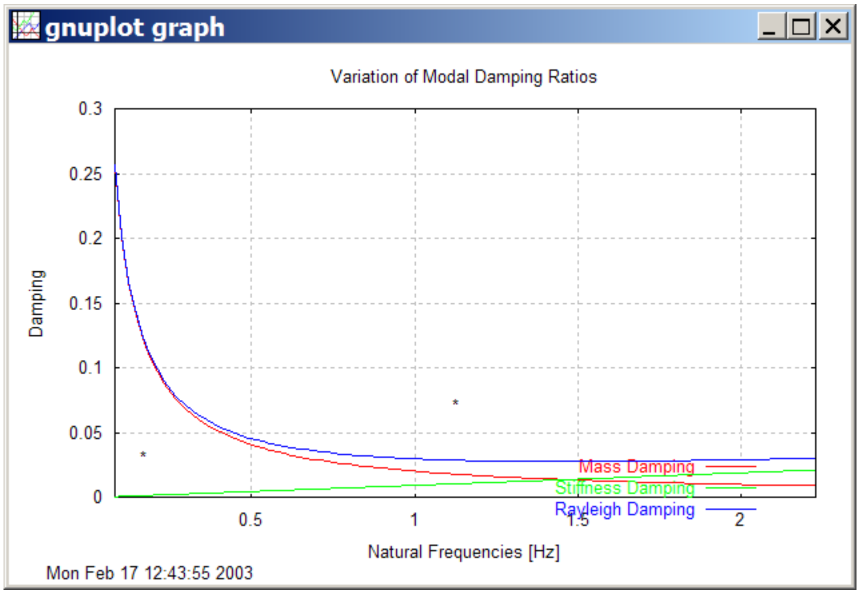
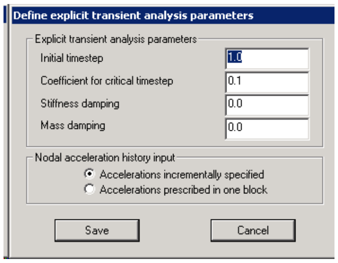


Figure 6.13: Computed Rayleigh Damping Coefficients



Explicit transient analysis parameters

Initial timestep: 1.0

Coefficient for critical timestep: 0.1

Stiffness damping: 0.0

Mass damping: 0.0

Nodal acceleration history input

● Accelerations incrementally specified

○ Accelerations prescribed in one block

Figure 6.14: Data Entry for Explicit Transient Analysis

6.2.3 Strain Smoothing Options

Merlin: 1.5

StrainSmooth instructs the computer to 'improve' the values for nodal strain through iterations using the technique of Zienkiewicz, Kui, and Nakazawa. No inversion of the consistent projection matrix is required (thus, the smoothing procedure is computationally inexpensive).

C-Splitting instructs the computer to 'improve' the values for nodal strain through iterations. Very similar to StrainSmooth, but C-Splitting converges faster.

Default instructs the computer to calculate strain with a lumped projection matrix.

6.2.4 Fracture Mechanics Options

Merlin: 1.4

LEFM sets a linear elastic fracture mechanics analysis for use in stress analysis only.

> **J-Integral** specifies the J-integral (Rice, Hellen, and Blackburn) to be used to indirectly calculate stress intensity factors from energy quantities.
>
> **S-Integral** specifies the S-integral (Stern and Hong) to directly compute the stress intensity factors based on reciprocal work.
>
> **B-Integral** specifies the B-integral (Babuska and Miller) for calculation of stress intensity factors based on energy release.
>
> **V-Integral** Specifies that the volume integral (based on J) is to be used to determine the SIF in 3D.
>
> **ContourPath Radius** defines the radius of the contour around the crack tip (2D) or the crack front (3D) from which the J/B or V integrals will be determined.

NLFM For nonlinear fracture mechanics (based on the cohesive crack model which requires the insertion of icm elements along the crack.

6.2.5 Print Options

Merlin: 1.7

PrintAAR prints incremental AAR strains.

PrintAccel prints nodal accelerations at each increment.

PrintConVar prints nodal constraint variables from stress analysis to output file.

PrintCrack prints crack displacement, pressure, and stress profiles from fracture mechanics.

PrintDisp user to specify the nodal displacements for stress analyses are printed to the output file.

PrintDynamic prints in an external file results of dynamic analysis, check.

PrintEigen prints results of the eigenvalue analysis.

PrintErrEst prints error estimates for strain energy and strain resulting from a stress analysis.

PrintReact user specifies the nodal reactions for a stress, heat flow or seepage flow analysis are printed to the output file.

PrintReinf prints embedded reinforcement information to the output file.

PrintResid prints residuals from stress, heat or seepage analysis to output file.

PrintState prints state variables from stress analysis to output file.

PrintStrain prints nodal strains from a stress analysis to the output file.

PrintStrain (GP) allows user to print Gauss points strains from a stress analysis to the output file.

PrintStress prints nodal stresses from a stress analysis to the output file.

PrintStress (GP) allows user to print Gauss points stresses from a stress analysis to the output file.

PrintTemp prints nodal temperatures to output file following heat transfer analysis.

PrintHead prints nodal heads for a seepage flow problem to output file.

PrintPress prints nodal heads in form of pressures to output file following seepage analysis.

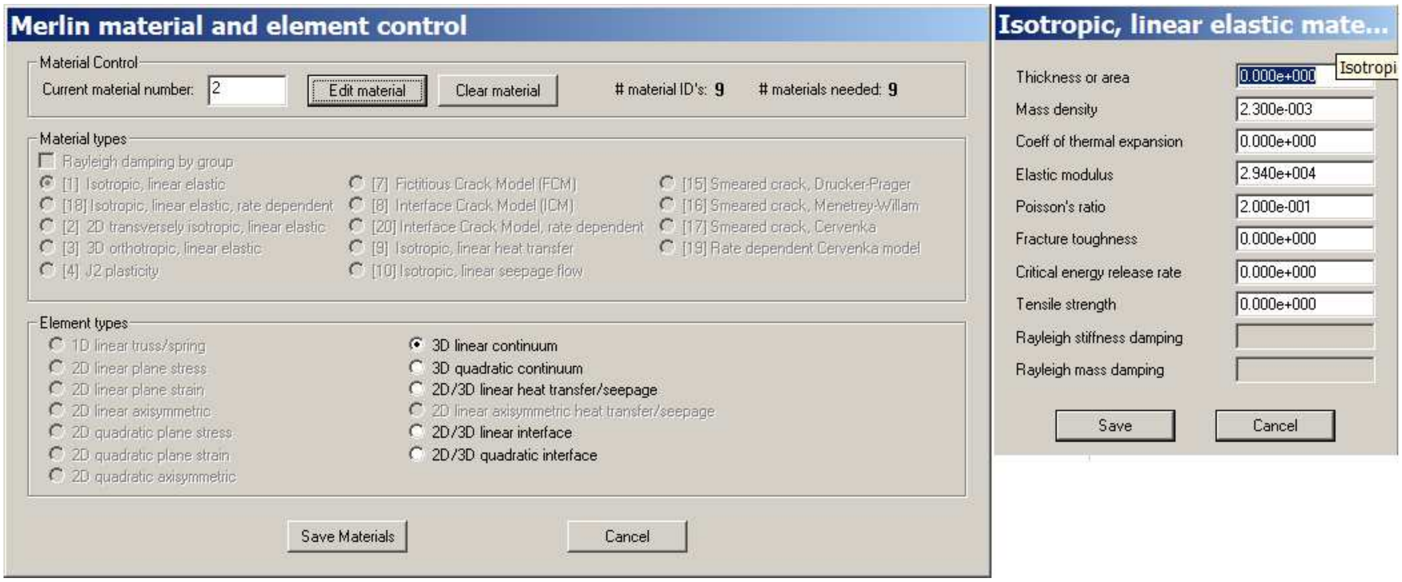
PrintFlux prints nodal fluxes to output file following heat transfer or seepage flow analysis.

Print_t_sif prints mode I stress intensity factors for each time increment of dynamic crack propagation analysis. This option must be used together with Transient, Impact, and LEFM.

PrintVeloc prints nodal velocities.

6.3 Material and Element Groups

KumoNoSu keeps track of: a) the number of material and element which must be defined, and b) if the problem is 2D or 3D. Hence, when the Material tag is selected, the user is prompted to identify those, Fig. 6.15.



Merlin material and element control

Material Control
Current material number: 2 | Edit material | Clear material | # material ID's: 9 | # materials needed: 9

Material types
□ Rayleigh damping by group
● [1] Isotropic, linear elastic
○ [18] Isotropic, linear elastic, rate dependent
○ [2] 2D transversely isotropic, linear elastic
○ [3] 3D orthotropic, linear elastic
○ [4] J2 plasticity
○ [7] Fictitious Crack Model (FCM)
○ [8] Interface Crack Model (ICM)
○ [20] Interface Crack Model, rate dependent
○ [9] Isotropic, linear heat transfer
○ [10] Isotropic, linear seepage flow
○ [15] Smeared crack, Drucker-Prager
○ [16] Smeared crack, Menetrey-Willam
○ [17] Smeared crack, Cervenka
○ [19] Rate dependent Cervenka model

Element types
○ 1D linear truss/spring
○ 2D linear plane stress
○ 2D linear plane strain
○ 2D linear axisymmetric
○ 2D quadratic plane stress
○ 2D quadratic plane strain
○ 2D quadratic axisymmetric
● 3D linear continuum
○ 3D quadratic continuum
○ 2D/3D linear heat transfer/seepage
○ 2D linear axisymmetric heat transfer/seepage
○ 2D/3D linear interface
○ 2D/3D quadratic interface

Save Materials | Cancel

Isotropic, linear elastic mate...

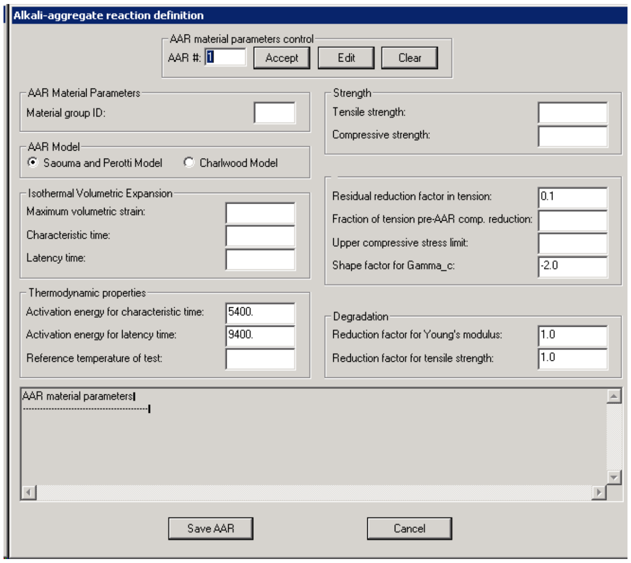
Thickness or area	0.000e+000
Mass density	2.300e-003
Coeff of thermal expansion	0.000e+000
Elastic modulus	2.940e+004
Poisson's ratio	2.000e-001
Fracture toughness	0.000e+000
Critical energy release rate	0.000e+000
Tensile strength	0.000e+000
Rayleigh stiffness damping	
Rayleigh mass damping	

Figure 6.15: Material Input Data

Element Type This is done by indicating simply the type of element (only those possible in the context of the current analysis are enabled, the others are greyed out). Merlin: 2.2.2

The corresponding Constitutive Model. Again, only those possible are enabled. Merlin: 2.2.3

Once a material type has been selected, KumoNoSu will then pop-up a second window inside which the user will enter the corresponding material properties. Whereas more details about these can be found in the Merlin manual, the sections below describe the data to be specified for each material type.


Alkali-aggregate reaction definition

AAR material parameters control
AAR #: 1

AAR Material Parameters
Material group ID:

AAR Model
● Saouma and Perotti Model ○ Charlwood Model

Isothermal Volumetric Expansion
Maximum volumetric strain:
Characteristic time:
Latency time:

Thermodynamic properties
Activation energy for characteristic time: 5400.
Activation energy for latency time: 9400.
Reference temperature of test:

Strength
Tensile strength:
Compressive strength:

Residual reduction factor in tension: 0.1
Fraction of tension pre-AAR comp. reduction:
Upper compressive stress limit:
Shape factor for Gamma_c: -2.0

Degradation
Reduction factor for Young's modulus: 1.0
Reduction factor for tensile strength: 1.0

AAR material parameters

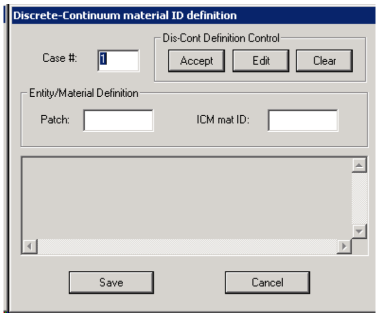
Figure 6.16: AAR User Interface

6.4 AAR

Merlin: 2.2.4

6.5 Discrete/Continuum Groups

Merlin: 2.2.3.5


Discrete-Continuum material ID definition

Dis-Cont Definition Control
Case #: 1

Entity/Material Definition
Patch: ICM mat ID:

Figure 6.17: Discrete ContinuumUser Interface

6.6 Eigenmode

Merlin: 3.4.1.9.1

Eigenvalue analysis is also possible, Fig. 6.18. There are two choices:

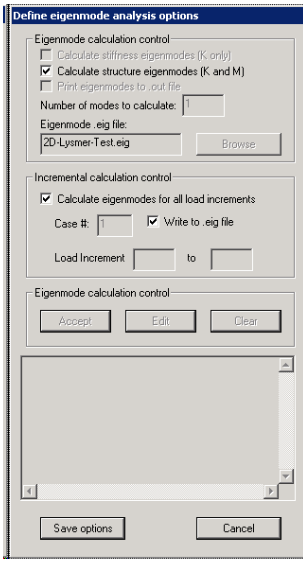
1. Natural frequencies and mode shapes of the structures determined from

$$(\mathbf{K} - \lambda\mathbf{M})\mathbf{u} = 0 \tag{6.1}$$

2. Eigenmodes (and mode shapes) of the stiffness matrix only.

$$(\mathbf{K} - \lambda\mathbf{I})\mathbf{u} = 0 \tag{6.2}$$

   this is an important feature when a nonlinear analysis is performed, as it enables the analyst to check the number of zero (or the decrease of the lowest ones) eigenvalues as damage progresses inside the structure. Eigenmodes can be computed either for all increments or for selected ones.



Define eigenmode analysis options

Eigenmode calculation control

□ Calculate stiffness eigenmodes (K only)

☑ Calculate structure eigenmodes (K and M)

□ Print eigenmodes to .out file

Number of modes to calculate: 1

Eigenmode .eig file: 2D-Lysmer-Test.eig

Incremental calculation control

☑ Calculate eigenmodes for all load increments

Case #: 1 ☑ Write to .eig file

Load Increment to

Eigenmode calculation control

Figure 6.18: Eigenmode Specification

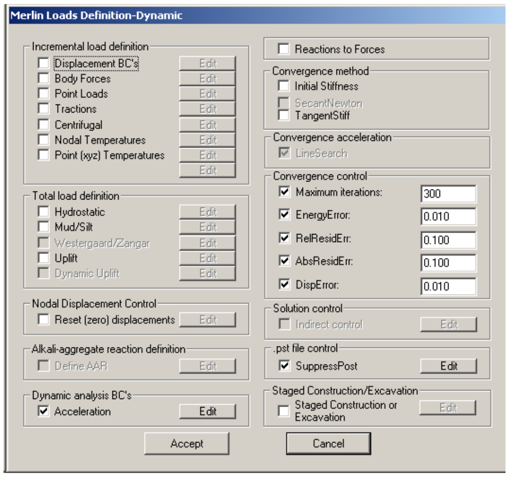
In both cases, the user must specify the **number of eigenmodes** to be determined (the larger the number, the longer the CPU time)

Results can be printed on a separate file, selected by **browse**. This file in turn can be viewed by `Spider` which will display and animate the mode shapes.

6.7 Loads

The load group is divided into two parts: Incremental and Total. The Incremental group of loads is a load applied only for the specified increments. On the other hand the Total load, once applied remains in effect for all subsequent increments.

A note on



Merlin Loads Definition-Dynamic

Incremental load definition

□ Displacement BC's

□ Body Forces

□ Point Loads

□ Tractions

□ Centrifugal

□ Nodal Temperatures

□ Point (xyz) Temperatures

Total load definition

□ Hydrostatic

□ Mud/Silt

□ Westergaard/Zangar

□ Uplift

□ Dynamic Uplift

Nodal Displacement Control

□ Reset (zero) displacements

Alkali-aggregate reaction definition

□ Define AAR

Dynamic analysis BC's

☑ Acceleration

□ Reactions to Forces

Convergence method

□ Initial Stiffness

□ SecantNewton

□ TangentStiff

Convergence acceleration

☑ LineSearch

Convergence control

☑ Maximum iterations: 300

☑ EnergyError: 0.010

☑ RelResidErr: 0.100

☑ AbsResidErr: 0.100

☑ DispError: 0.010

Solution control

□ Indirect control

.pst file control

☑ SuppressPost

Staged Construction/Excavation

□ Staged Construction or Excavation

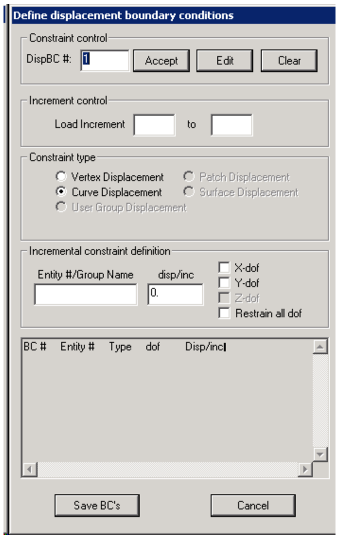
Figure 6.19: Loads Type

6.7.1 Incremental Load Definition

6.7.1.1 Displacement BC's

Merlin: 3.4.1.3

Merlin considers the boundary conditions as an intrinsic part of the load. Boundary conditions can be specified for individual vertices, curve, patch or surface, Fig. 6.20. The followinig must be specified:


Define displacement boundary conditions

Constraint control

DispBC #: 1

Increment control

Load Increment ____ to ____

Constraint type

○ Vertex Displacement ○ Patch Displacement
● Curve Displacement ○ Surface Displacement
○ User Group Displacement

Incremental constraint definition

Entity #/Group Name ____ disp/inc 0.

□ X-dof
□ Y-dof
□ Z-dof
□ Restrain all dof

BC #	Entity #	Type	dof	Disp/inc

Figure 6.20:

Load increment(s) Define load increment or increment interval for current bc. This enables the user to specify the same boundary conditions for some or all the increments.

Constraint type Entity type to be restrained (vertex or curve in 2D; vertex, curve, patch, or surface in 3D). If the selected entity type has minor entities associated to it (such as the end vertices for a curve, or curves to a surface/patch), those minor entities may be restrained also by selecting **Restrain associated minor entities**.

DispBC # Number of the current bc (for reference only).

Entity # Number of the entity to be restrained

DOF # Degree of freedom to be restrained If every dof of the current entity is to be restrained, select Restrain all dof to do so

disp/inc Displacement magnitude per increment (usually zero which is the default value).

6.7.1.2 Body Forces

Merlin: 3.4.1.8

The body forces are defined by specifying the acceleration of gravity to be applied to all or selected material group. Each group in turn would have had its mass defined separately. Hence, we can have a material with a mass (thus subjected to an acceleration force), but without gravity load (such as the rock foundation in a dynamic analysis of a dam), Fig. 6.21.

Load # Number of the current load (for reference).

Load increment(s) Two possible cases:

1. If all materials are subjected to body force, only specify the increment in which body forces are first required. If only select materials are subjected to body force, specify the beginning nd end load increments for this material group list.

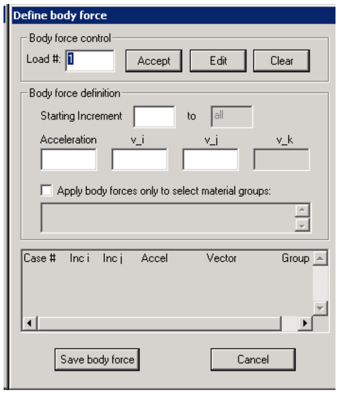


Figure 6.21: Definition of Body Forces

Acceleration Magnitude of the acceleration applied to the materials.

v_i, v_j, (v_k) Components of the vector defining the direction. of acceleration.

Select groups If option 2 is required, check the box and defined the select material group numbers that are subjected to body forces. Careful this is for material group and not regions or patches.

6.7.1.3 Point Loads

Merlin: 3.4.1.4

Fig. 6.22.

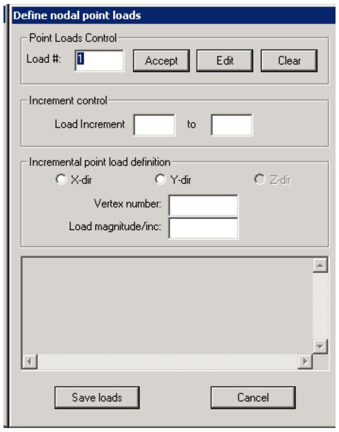


Figure 6.22: Point Load Definition

Load increment(s) Define load increment or increment interval for current load. Remember that this is an *incremental* load.

Load number Number of the current load (for reference).

Vertex number Number of the vertex to be loaded.

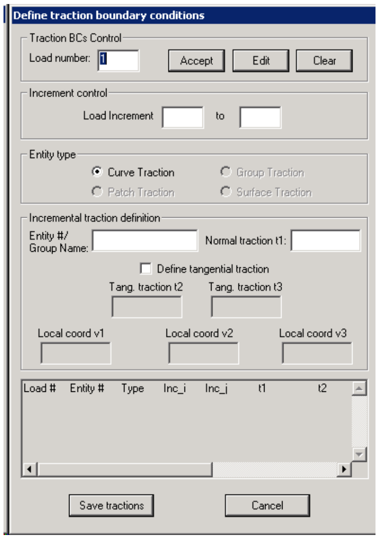
Load magnitude/inc Magnitude of the point load per load increment (ΔP).

Load direction X-dir, Y-dir, or Z-dir (3D only)

6.7.1.4 Tractions

Merlin: 3.4.1.5

Traction loads on curves, patches or surfaces is defined as follows, Fig. 6.23:



Load number: 1

● Curve Traction

Figure 6.23: Traction Load Definition

Load number number of the current load (for reference).

Load increment(s) Define load increment or increment interval for current load.

Entity type entity type to be loaded (2D: curve only; 3D: curve, patch, or surface).

Entity number number of loaded entity

Normal traction t_n traction load applied normal to the element surface (positive = tension, negative = compression).

Tang. Traction t_{n1}, t_{n2} tangential traction components in the local y-axis (t_{n1}) and z-axis (t_{n2}); t_{n2} traction only active for 3D models

Local coordinates v_1, v_2, v_3 Vector defining the local coordinate system for the element surface; only required for 3D models, Fig. 6.24.

Note that for 2D problems, the traction is defined for unit thickness, hence Merlin will internally multiply the thickness t.

6.7.1.5 Centrifugal

Merlin: 3.4.1.11

The `Centrifugal` option enables the user to define the incremental centrifugal forces for stress analyses. This option is to be used when the centrifugal (or gravity) forces varies along a structure (such as in tests performed inside a centrifuge, or electromagnetic forces), Fig. 6.25, and the force is given by

$$F = m\omega^2 r \tag{6.3}$$

where ω is the angular velocity (rad/s).

Load number Number of the current load (for reference).

Load increment(s) Define load increment or increment interval for current load.

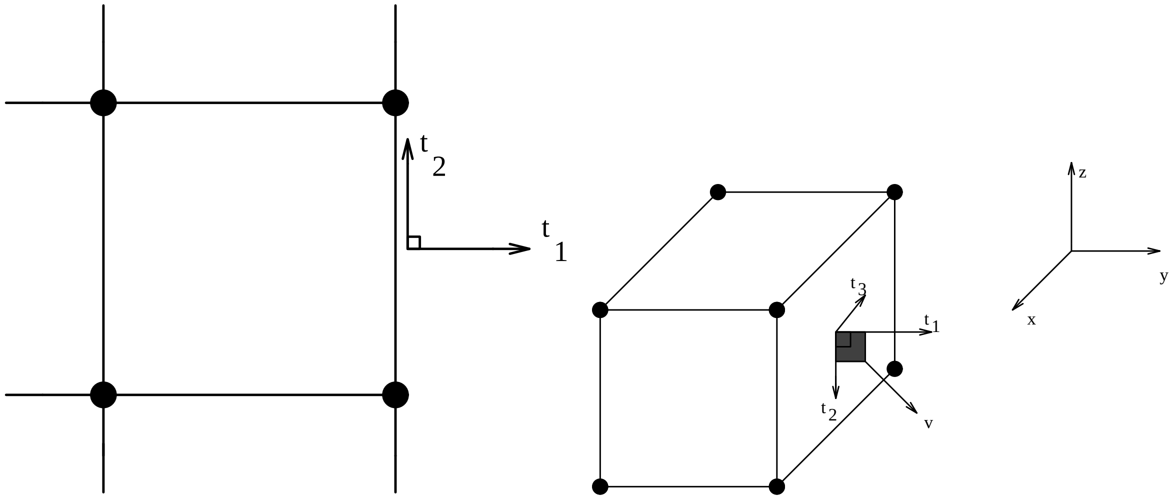
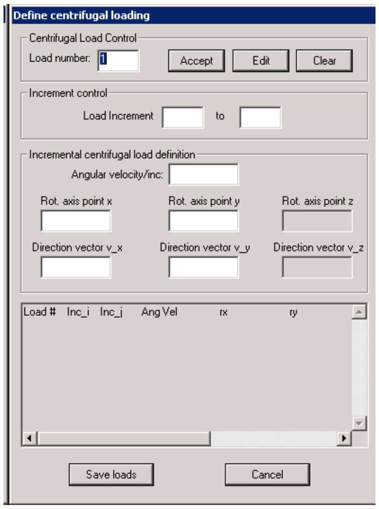


Figure 6.24: Direction of Traction load



Load number: 1

Figure 6.25: Centrifugal Load Definition

Angular velocity Magnitude of angular velocity ω per load increment.

Axis point x_i, x_j, x_k Coordinates of a point on the axis of rotation.

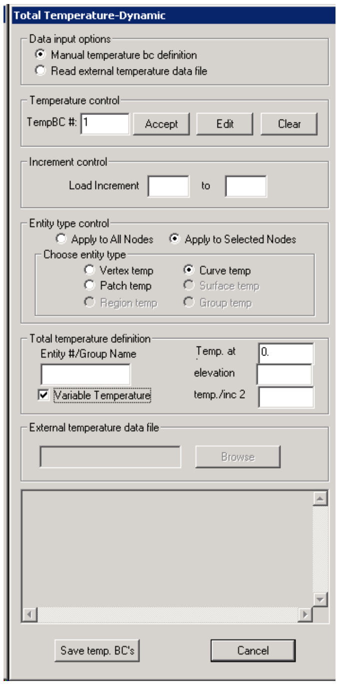
Direction vector v_i, v_j, v_k Vector specifying the direction of rotation.

Note: once defined, centrifugal loading is applied to the entire model.

6.7.1.6 Nodal Temperature

6.7.1.6.1 GUI Entry

Merlin: 3.4.1.13.1



Total Temperature-Dynamic

Data input options
● Manual temperature bc definition
○ Read external temperature data file

Temperature control
TempBC #: 1

Increment control
Load Increment ___ to ___

Entity type control
○ Apply to All Nodes ● Apply to Selected Nodes

Choose entity type
○ Vertex temp ● Curve temp
○ Patch temp ○ Surface temp
○ Region temp ○ Group temp

Total temperature definition
Entity #/Group Name ___
☑ Variable Temperature
Temp. at 0.
elevation ___
temp./inc 2 ___

External temperature data file ___

Figure 6.26: Nodal Temperature Load Definition

1. Must specify if temperatures are manually entered for each node, or through an external data file (see below).

2. Load increments in which the temperature is to be specified. Keep in mind that this is an *incremental* load.

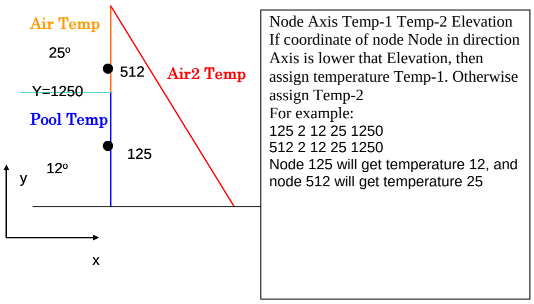
3. A user specified temperature can be applied to all the nodes (if we have a uniformm temperature), or to selected entity (vertex, curve, patch, surface or region).

4. User can specify

   One Temperature for all the nodes in the selected entity.

   Two Temperatures for all the nodes in the selected entity: a reference temperature if the (z) elevation is below a certain value, and another one if the z elevation is above the specified elevation. This is convenient for the upstream temperature of a dam, where part of it is submerged by water, and another is exposed to air, Fig. **??**.

Note: The GUI should be corrected to specify axis.



Node Axis Temp-1 Temp-2 Elevation
If coordinate of node Node in direction Axis is lower that Elevation, then assign temperature Temp-1. Otherwise assign Temp-2
For example:
125 2 12 25 1250
512 2 12 25 1250
Node 125 will get temperature 12, and node 512 will get temperature 25

Figure 6.27: Definition of Variable Nodal Temperatures

6.7.1.6.2 External File

To facilitate data entry when a large number of nodal temperatures must be defined, Kumo allows the definition of nodal temperature through an external file.

The external file should have an extension `.dat` and have the following format:

- `Temperature` Keyword to specify hydrostatic load.
- # Comment if applicable
- `NB` Number of blocks in the file (integer). A block is a group of entities (vertices, curves, patches, surfaces, or regions) having the same incremental hydrostatic loads, such as upstream and downstream faces.
- For each block:
  - `Block` # (i.e the file should include the string `Block` followed by an integer.
  - `Vertex|Curve|Patch|Surface|Region`Entity numbers subjected to hydrostatic load. One line for Curve (if applicable), one for Patch (if applicable), and one for Surface (if applicable). If there are too many entries for a given entities, use multiple lines each beginning with a string equal to the entity type (such as `Surface` followed by integers.
  - `Increment` #N (i.e the string increment) followed by the total number of load increments for which we will be defining a hydrostatic load.
  - For each increment along one line:
    - Increment number (integer)
    - Temperature corresponding to that increment (incremental).

6.7.1.7 Point (xyz) Temperatures

Merlin: 3.4.1.13.2

6.7.1.7.1 GUI Specification

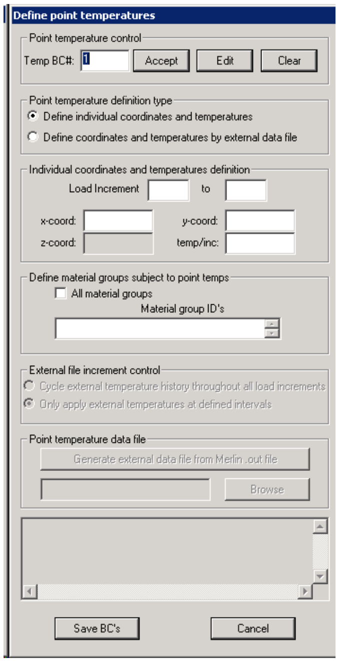
Careful temperature is incremental.

6.7.1.7.2 External File Input

To facilitate data entry when a large number of point temperatures must be defined, Kumo allows the definition of point temperaturs through an external file.

The external file should have an extension `.dat` and have the following format:

- First Line:
  - Number of increments.
  - First increment number in Merlin to load this file
  - Last increment number in Merlin to load this file.



Define point temperatures

Point temperature control

Temp BC#: 1

Point temperature definition type

● Define individual coordinates and temperatures

○ Define coordinates and temperatures by external data file

Individual coordinates and temperatures definition

Load Increment ____ to ____

x-coord: ____ y-coord: ____

z-coord: ____ temp/inc: ____

Define material groups subject to point temps

□ All material groups

Material group ID's

External file increment control

○ Cycle external temperature history throughout all load increments

● Only apply external temperatures at defined intervals

Point temperature data file

Figure 6.28: Point Temperature Load Definition

- Number of point temperatures.
- Spatial dimension (2 or 3).

- For each point temperature, specify on one line:
  - id number (integer starting with 1).
  - x,y,[z] coordinates.
- For each increment, specify on one line
  - id number of point
  - Temperature of the point. Note temperatures are incremental, hence the first increment defines the base temperature, and the second increment specifies the temperature increase/decrease with respect to the previous one. Hence for any increment n, the temperature is equal to $T_1 + \Sigma_{i=2}^{i=n} T_i$.

6.7.2 Total Load Definition

Total loads are those which once applied, remain constant in all subsequent load increments.

6.7.2.1 Hydrostatic

6.7.2.1.1 GUI Entries

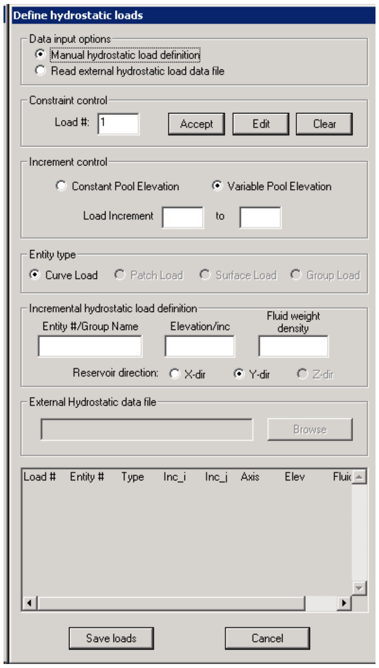
Merlin: 3.4.1.7.1

Hydrostatic load is defined by Fig. 6.29.

Load increment(s) Define load increment or increment interval for current load.

Entity type Type of loaded entity (curve in 2D, patch or surface in 3D).

Load number Number of the current load (for reference).



Define hydrostatic loads

Data input options
● Manual hydrostatic load definition
○ Read external hydrostatic load data file

Constraint control
Load #: 1

Increment control
○ Constant Pool Elevation ● Variable Pool Elevation
Load Increment ___ to ___

Entity type
● Curve Load ○ Patch Load ○ Surface Load ○ Group Load

Incremental hydrostatic load definition
Entity #/Group Name ___ Elevation/inc ___ Fluid weight density ___
Reservoir direction: ○ X-dir ● Y-dir ○ Z-dir

External Hydrostatic data file

Load #	Entity #	Type	Inc_i	Inc_j	Axis	Elev	Fluid

Figure 6.29: Hydrostatic Load Definition

Entity # Number of the loaded entity.

Elevation/inc Fluid elevation change per load increment (Δh)

Fluid weight Fluid weight density (force/volume).

Direction of fluid height in global coordinates, X-dir, Y-dir, or Z-dir (3D only).

Note that in general, a first load number must be defined which brings the water elevation to the base of the dam. Then, incremental water height (Δh) can be defined to raise the water at each increment.

6.7.2.1.2 External File

To facilitate data entry when a large number of hydrostatic loads must be defined, Kumo allows the definition of hydrostatic load through an external file.

The external file should have an extension `.dat` and have the following format:

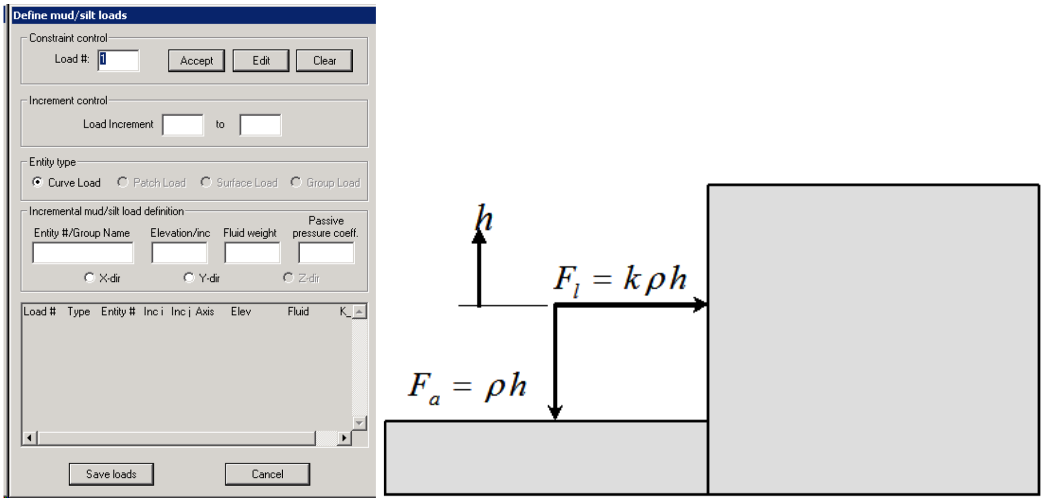
- `Hydrostatic` Keyword to specify hydrostatic load.
- # Comment if applicable
- `NB` Number of blocks in the file (integer). A block is a group of entities (curves, patches or surfaces) having the same incremental hydrostatic loads, such as upstream and downstream faces.
- For each block:
  - `Block` # (i.e the file should include the string `Block` followed by an integer.
  - `Curve|Patch|Surface` Entity numbers subjected to hydrostatic load. One line for Curve (if applicable), one for Patch (if applicable), and one for Surface (if applicable). If there are too many entries for a given entities, use multiple lines each beginning with a string equal to the entity type (such as `Surface` followed by integers.
  - `Increment` #N (i.e the string increment) followed by the total number of load increments for which we will be defining a hydrostatic load.

- For each increment, and along the same line:
    - ∗ Increment number (integer)
    - ∗ Δ Elevation of the reservoir. Note that this is the incremental elevation and not the total elevation. Hence in the first increment we (typically) bring the pool elevation to the base elevation of the dam (hence the dam is not loaded), and in subsequent increments we simply specify by how much we raise (+ve) or lower (-ve) the pool.
    - ∗ γ_w Fluid weight density
    - ∗ Axis orientation (2 If fluid is along y, 1 if along the x)

6.7.2.2 Mud/Silt

Mud and silt load is sightly different than hydrostatic load, as the lateral pressure is equal to the vertical pressure times the passive coefficient k. For an inclined curve (2D) or patch (3D), KumoNoSu will automatically calculate the mud/silt pressure normal to the surface, given the vertical and lateral components F_a and F_l.

It is defined, as follows, Fig. 6.30.:



Define mud/silt loads

Constraint control

Load #: 1

Incremental control

Load Increment ___ to ___

Entity type

● Curve Load ○ Patch Load ○ Surface Load ○ Group Load

Incremental mud/silt load definition

Entity #/Group Name | Elevation/inc | Fluid weight | Passive pressure coeff.

○ X-dir ○ Y-dir ○ Z-dir

Load # Type Entity # Inc i Inc j Axis Elev Fluid K_

Figure 6.30: Mud Silt Definition

Load increment(s) Define load increment or increment interval for current load.

Entity type Type of loaded entity (curve in 2D, patch or surface in 3D).

Load number Number of the current load (for reference).

Entity # Number of the loaded entity.

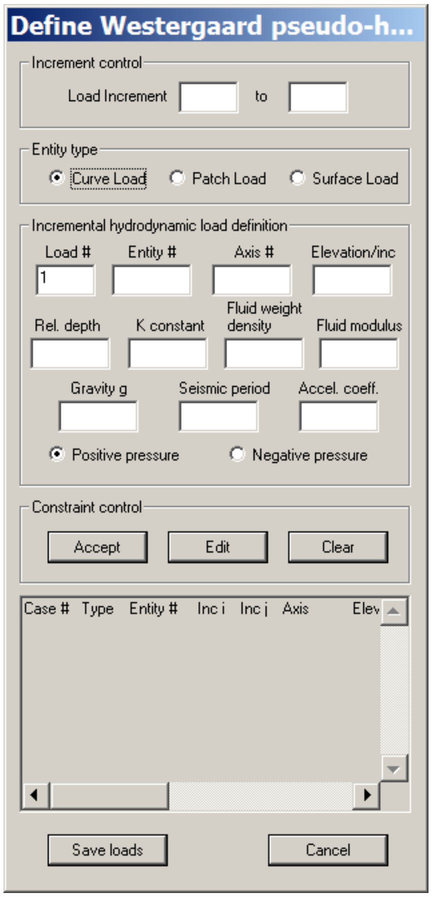
Elevation/inc Fluid elevation change per load increment (Δh).

Fluid weight Fluid weight density (force/volume).

Passive pressure coeff. Passive pressure coefficient for lateral pressure due to mud/silt (k coefficient).

Direction of fluid height in global coordinates, X-dir, Y-dir, or Z-dir (3D only).

As for hydrostatic load, a first load number must be defined which brings the mud elevation to the base of the dam. Then, incremental mud height (Δh) can be defined to raise the mud at each increment.



Define Westergaard pseudo-h...

Increment control

Load Increment ______ to ______

Entity type

● Curve Load ○ Patch Load ○ Surface Load

Incremental hydrodynamic load definition

Load #	Entity #	Axis #	Elevation/inc
1			

Rel. depth	K constant	Fluid weight density	Fluid modulus

Gravity g	Seismic period	Accel. coeff.

● Positive pressure ○ Negative pressure

Constraint control

Case #	Type	Entity #	Inc i	Inc j	Axis	Elev

Figure 6.31: Westergaards Added Mass Load Definition

6.7.2.3 Westergaard

Merlin: 3.4.1.7.3 3.4.1.7.3

Westerggard's pseudo-hydrodynamic added masses are determined as follows, Fig. 6.31:

Load increment(s) Define load increment or increment interval for current load.

Entity type Type of loaded entity (curve in 2D, patch or surface in 3D).

Load # Number of current load (for reference).

Entity # number of the loaded entity.

Axis # Axis defining reservoir depth: 1 = X, 2 = Y, 3 = Z (3D only).

Elevation/inc Elevation of the reservoir surface per load increment note that this is not the relative depth of the reservoir, but rather the elevation of the surface).

Rel. depth Relative depth of the reservoir.

K constant : K constant for Westergaard equation (defined by Westergaard as 51.0 lb/ft^3 or 8,011.4 N/m^3 for water).

Fluid weight Weight per unit volume of the fluid.

Fluid modulus Elastic modulus of the fluid (may be taken as 300 kips/in^2 or 2.068 GPa for water).

Gravity g Acceleration of gravity.

Seismic period Period of the earthquake motion.

Accel coeff Lateral acceleration as a fraction of gravity.

Pressure type Positive (towards dam) or negative.

Note that if the earthquake excitation is not along the stream axis, than an orthogonal model of the added mass must be specified, Fig. 6.32:

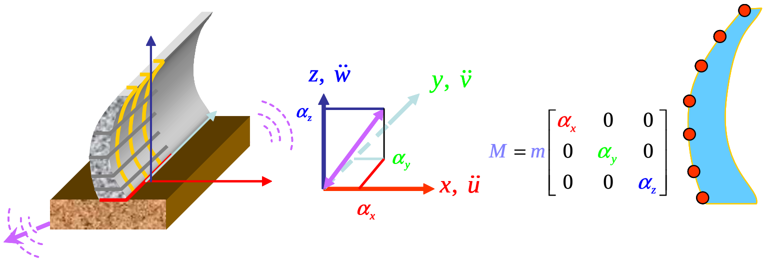

z, $\ddot{w}$
y, $\ddot{v}$
x, $\ddot{u}$
α_z
α_y
α_x
$M = m\begin{bmatrix}\alpha_x & 0 & 0\\ 0 & \alpha_y & 0\\ 0 & 0 & \alpha_z\end{bmatrix}$

Figure 6.32: Westergaards Orhtogonal Added Mass Load Definition

6.7.2.4 Uplift

6.7.2.4.1 GUI Definition

Merlin: 3.4.1.7.4

Two uplift models are implemented in KumoNoSu :

Full is when we have an uplift pressure, only along a crack, equal to the upstream hydrostatic pressure. There is no uplift pressure from the crack tip to the downstream face.

Ferc is identical to Full, however there is a linearly varying pressure from the crack tip to the tailwater hydrostatic pressure, Fig. 6.33.

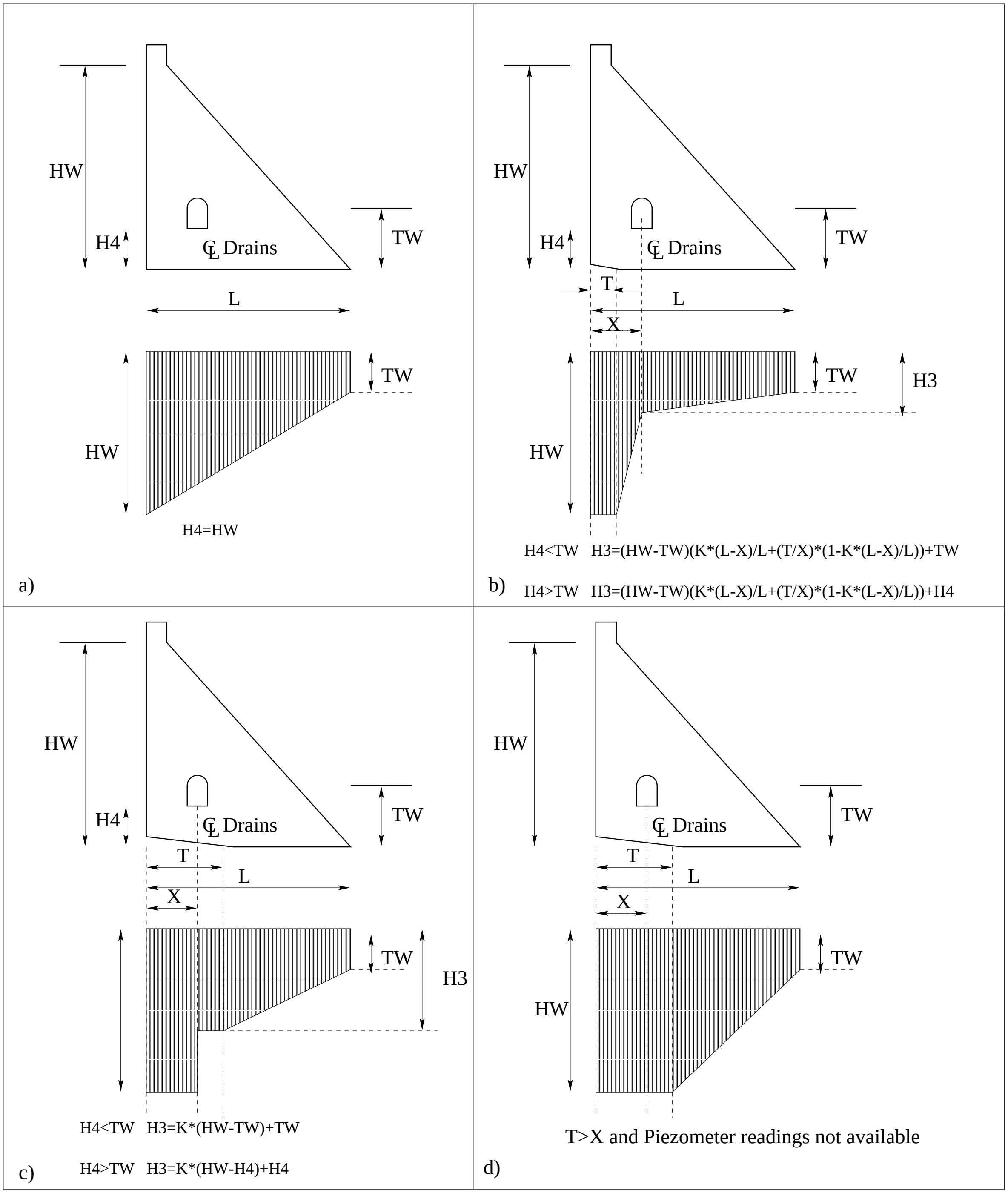
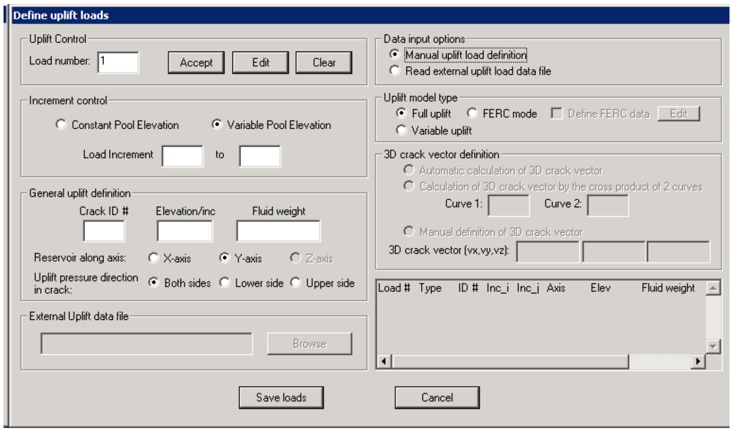

HW
H4
℄ Drains
TW
L
TW
HW
H4=HW
a)
HW
H4
℄ Drains
TW
T
L
X
TW
H3
HW
H4<TW H3=(HW-TW)(K*(L-X)/L+(T/X)*(1-K*(L-X)/L))+TW
b)
H4>TW H3=(HW-TW)(K*(L-X)/L+(T/X)*(1-K*(L-X)/L))+H4
HW
H4
℄ Drains
TW
T
L
X
TW
H3
H4<TW H3=K*(HW-TW)+TW
c)
H4>TW H3=K*(HW-H4)+H4
HW
℄ Drains
TW
T
L
X
TW
HW
T>X and Piezometer readings not available
d)

Figure 6.33: FERC Uplift Loads


Define uplift loads
Uplift Control
Accept
Edit
Clear
Increment control
○ Constant Pool Elevation
Load Increment
to
General uplift definition
Crack ID #
Elevation/inc
Fluid weight
Reservoir along axis:
○ X-axis
○ Z-axis
Uplift pressure direction in crack:
○ Lower side
○ Upper side
External Uplift data file
Browse
Data input options
○ Read external uplift load data file
Uplift model type
○ FERC mode
□ Define FERC data
Edit
○ Variable uplift
3D crack vector definition
○ Automatic calculation of 3D crack vector
○ Calculation of 3D crack vector by the cross product of 2 curves
Curve 1:
Curve 2:
○ Manual definition of 3D crack vector
3D crack vector (vx,vy,vz):
Load # Type ID # Inc_i Inc_j Axis Elev Fluid weight
Save loads
Cancel

Load number: 1

● Variable Pool Elevation

● Y-axis

● Both sides

● Manual uplift load definition

● Full uplift

Figure 6.34: Uplift Load Definition

KumoNoSu defines the uplift as follows, Fig. 6.34:

Load increment(s) Define load increment or increment interval for current load.

Uplift model type Full uplift or the FERC model.

Load # Number of the current load (for reference).

Crack Path # Number of the crack path subjected to uplift pressures.

Axis # Axis defining reservoir depth: 1 = X, 2 = Y, 3 = Z (3D only).

Elevation/inc Fluid elevation per load increment.

Fluid weight Weight per unit volume of the fluid.

Crack prop axis # Global axis which defines the direction of crack propagation (necessary for 3D uplift only).

Crack surfaces subjected to uplift : Both sides, lower surface only, or upper surface only.

If the FERC model is used, Fig.6.33, then additional data must be entered:

Distance from crack mouth to the drain This is an average distance.

Drain efficiency e ($0 < e < 1$) where 1. corresponds to full efficiency)

- Before it is intersected by the crack. If there is no drain, specify 0.
- After it has been intersected by the crack.

Drain elevation Vertical distance between the crack subjected to uplift and the drain

Tail water elevation (or water elevation at the end of the crack).

Total base length of the dam, i.e from upstream to downstream. Note this is not the projected length, but the actual length of the crack. For multiple (inclined) segments, the total “curvilinear” length must be specified. If the base length is variable, then it would be more conservative to adopt the maximum length.

6.7.2.4.2 External File

To facilitate data entry when a large number of uplift loads must be defined, Kumo allows the definition of uplift load through an external file.

The external file should have an extension `.dat` and have the following format:

- `Uplift` Keyword to specify hydrostatic load.
- # Comment if applicable
- `NB` Number of blocks in the file (integer). A block is a group of cracks having the same uplift load.
- For each block:
  - `Block` # (i.e the file should include the string `Block` followed by an integer.
  - `Crack` followed by the cracks defining this particular block.
  - `Increment` #N (i.e the string increment) followed by the total number of load increments for which we will be defining an uplift load.
  - For each increment, and along the same line:
    - Increment number (integer)
    - Water elevation in global coordinate system. Note that this is an incremental uplift to be defined with respect to the previous total uplift pressure. Hence the very first increment should bring the uplift pressure to the dam reference elevation.
    - Unit weight of water, γ_W.
    - Id of the global coordinate axis along which the uplift is applied (x =1, y=2, z=3).
    - Directions of uplift forces. 0: both sides, -1: lower side, 1: upper side.
    - Direction of crack (from upstream to downstream): a) 2D: User must simply define direction of ±1 if along $\pm x$, and ±2 if along $\pm y$. b) or 3D: User must define a vector v_x, v_y, v_z from upstream to downstream direction.
    - If FERC Model is specified, continue on the same line by inserting
      - `FERC`
      - Distance from crack mouth to the drain
      - Drain efficiency e ($0 < e < 1$) where 1. corresponds to full efficiency) before it is intersected by the crack. If there is no drain, specify 0.
      - Drain efficiency after it has been intersected by the crack
      - Vertical distance between the crack subjected to uplift and the drain
      - Tail water elevation (or water elevation at the end of the crack) measured with respect to the crack subjected to uplift. Note that this is NOT incremental, but the total elevation.
      - Total base length of the dam from upstream to downstream. In 2D:User must specify the length. in 3D: If left as 0., then Merlin internally computes this length, if non-zero, then Merlin will consider this value as the length. Note that this is not the projected length, but the actual "curvilinear" length of the crack.

6.7.2.5 Dynamic Uplift

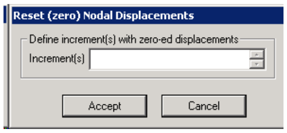
to insert

Merlin: 3.4.1.7.6

6.7.3 Reset Nodal Displacements

Merlin: 3.1.2

This enables the user to rest the nodal displacements (but not the stresses or the state variables) to zero after a certain increment, Fig. 6.35.



Reset (zero) Nodal Displacements

Define increment(s) with zero-ed displacements

Increment(s)

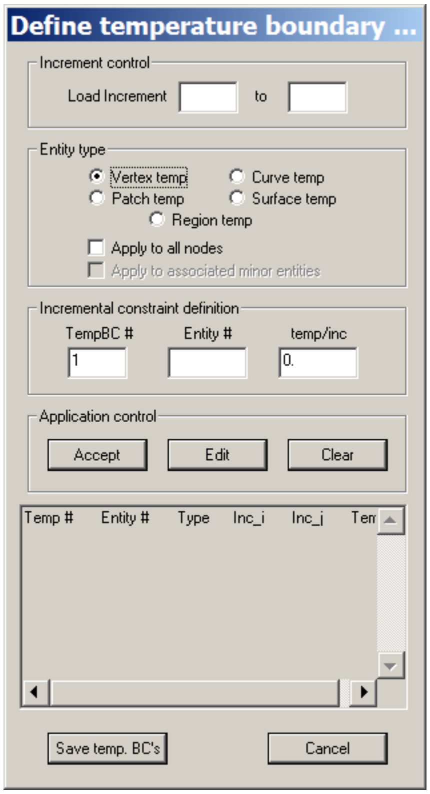
Figure 6.35: Reset (zero) Displacments Definition

6.7.4 Heat Transfer/Seepage Loads

MUST BE COMPLETED, CHECK POWER POINT FILES

6.7.4.1 Temperature

Temperature nodal loads can be specified not only for thermal effects but also for any other physical phenomena causing expansion (such as Alkali-Silica Reactions, in conjunction with a pseudo coefficient of thermal expansion defined in the material properties), Fig. 6.36.



Define temperature boundary ...

Increment control

Load Increment to

Entity type

● Vertex temp ○ Curve temp
○ Patch temp ○ Surface temp
○ Region temp

□ Apply to all nodes
□ Apply to associated minor entities

Incremental constraint definition

TempBC #	Entity #	temp/inc
1		0.

Application control

Temp #	Entity #	Type	Inc_i	Inc_j	Ten

Figure 6.36: Temperature Load Definition

Load increment(s) Define load increment or increment interval for current load.

Entity type Type of loaded entity (vertex or curve in 2D, all entity types in 3D). Note that:

- If every node in the FE mesh is to receive a temperature bc, select **Apply to all nodes**.
- If the selected entity type has minor entities associated to it (such as the end vertices for a curve), those minor entities may be included by selecting **Apply to associated minor entities**.

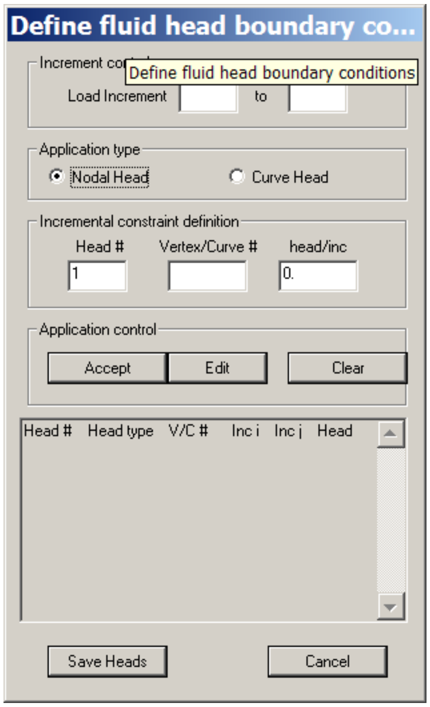
TempBC # Number of current bc (for reference).

Entity # Number of the entity receiving temperature bcs.

Temp/inc Temperature change (ΔT) per load increment.

6.7.4.2 Head

Head load is defined as follows, Fig. 6.37.



Define fluid head boundary co...

Define fluid head boundary conditions

Application type: ● Nodal Head ○ Curve Head

Head # 1

head/inc 0.

Figure 6.37: Head Load Definition

Load increment(s) Define load increment or increment interval for current load.

Entity type Type of loaded entity (vertex or curve).

Head # Number of current head(for reference).

Entity # Number of the entity receiving the head bcs.

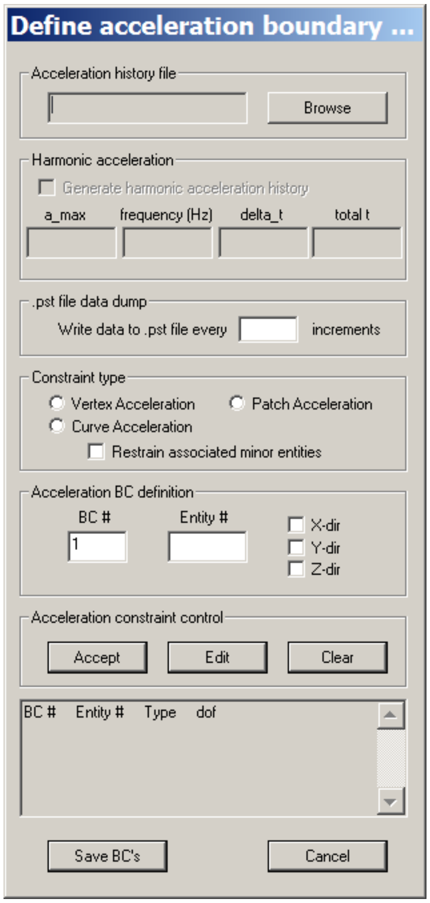
Head/inc Head per load increment.

6.7.5 Dynamic Analysis

This option is intended to be used for the application of an acceleration history for a dynamic analysis. Since the history is applied at load increment 1 of the dynamic analysis, no load increments or intervals may be specified, Fig. 6.38. Mandatory information:

Acceleration history file Name of text file containing the nodal acceleration history. This file should contain: $t_i, a_{xi}, a_{yi}, a_{zi}$.

.pst file data dump Increment interval at which to write analysis results to the `.pst` output file for viewing with Spider.



Define acceleration boundary ...

Acceleration history file

Harmonic acceleration

□ Generate harmonic acceleration history

a_max	frequency (Hz)	delta_t	total t

.pst file data dump

Write data to .pst file every ____ increments

Constraint type

○ Vertex Acceleration ○ Patch Acceleration
○ Curve Acceleration
□ Restrain associated minor entities

Acceleration BC definition

BC #: 1
Entity #:
□ X-dir
□ Y-dir
□ Z-dir

Acceleration constraint control

BC # Entity # Type dof

Figure 6.38: Acceleration Specification

Constraint type Entity type receiving acceleration bcs (vertex or curve in 2D, vertex, curve, or patch in 3D).

BC # Number of current acceleration bc (for reference).

Entity # Number of the entity to receive acceleration bcs.

DOF Degree of freedom to receive acceleration bcs.

6.7.5.1 Harmonic Excitation

In some applications, it is desirable to generate a harmonic excitation curve and apply it to the structure. If this option is selected, then the following additional data must be specified:

a_{max} Maximum acceleration magnitude.

frequency Frequency of the harmonic motion (Hz).

Δt Time step for the harmonic motion (may be overwritten by KumoNoSu if too large to capture the harmonic motion).

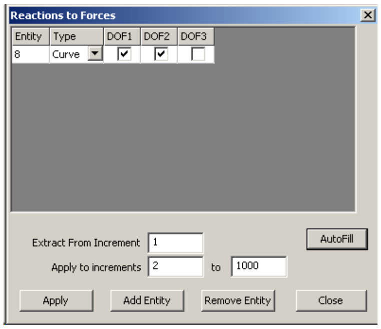
total t Total time of harmonic motion.

6.7.6 Reactions to Load

To identify the reactions which should be saved by Merlin and then applied as loads, through the Merlin Keyword `SaveReacts` and `Reacts2Loads`, user must invoke Reactions to load. Merlin: 3.1.3

Once selected, Fig. 6.39 the user must first specify the increment from which the reactions must be extracted (i.e. generate the `SaveReacts` command in Merlin). Then the user must also select those nodes and degrees of freedom with reactions which must be saved. Currently Kumo allows for the extraction of **all** the reactions only if the user selects `Auto-Fill`.

The user must then select the increment ranges (from-to) to which those reactions must be applied as nodal loads. Merlin: 3.1.4



Entity	Type	DOF1	DOF2	DOF3
8	Curve	☑	☑	□

Extract From Increment 1

Apply to increments 2 to 1000

Figure 6.39: Reactions to Load

6.7.7 Convergence Control

6.7.7.1 Convergence Control

Convergence control:

Maximum Iterations specify maximum number of iterations

Energy/Error specify relative convergence tolerance in terms of energy (recommended value product of displacement error and the absolute residual error). Merlin: 3.2.2

RelResidErr specify relative convergence tolerance in terms of residual forces (=10% , typically).

AbsResidErr specify absolute convergence tolerance in terms of residual forces. Especially useful for stress analysis involving thermal gradients (recommended value: 10%). Merlin: 3.2.3

DispError specify relative convergence tolerance in terms of displacements. (recommended value: 1%). Merlin: 3.2.5

6.7.7.2 Solution Method

KumoNoSu enables the user to select any one of the following: Merlin: 3.3

1. Initial Stiffness (which is the default mode for Merlin.
2. Secant Newton. Merlin: 3.3.1.2
3. Tangent Stiffness. Merlin: 3.3.1.1

and Line Search may also be specified.

6.7.7.3 Convergence Acceleration

Merlin: 3.3.2

6.7.8 Solution Control

If an indirect solution control (such as crack opening displacement) is selected, Fig. 6.40. The `SpecifyCOD` Merlin: 3.3.3

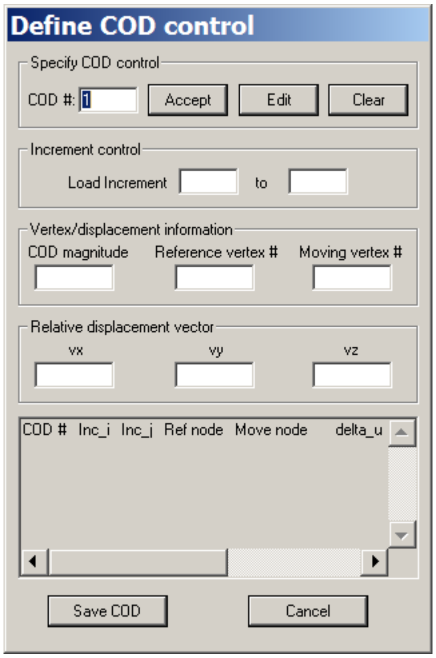


Figure 6.40: Indirect (COD) Solution Control

option enables the user to control the step size for the modified-Newton algorithm by specifying a relative displacement component between two nodes (de Borst 1986, de Borst 1987).

Load Increment from first and last load increment.

COD Magnitude The magnitude of the prescribed relative displacement component, $\Delta u'$. Note that this is not necessarily a crack opening displacement.

Reference vertex # reference node.

Moving vertex # second node.

vector components The three components v_i of the vector $\mathbf{v}$ defining the direction of the relative displacement component.

The magnitude of the prescribed relative displacement component and the direction vector components are floating point numbers and the node numbers are integers. The relative displacements (in the global coordinate system) are defined by subtracting the displacements for the "moving" node from the displacements for the reference node. The direction vector $\mathbf{v}$ is not required to be a unit vector; the program utilizes it before computing the relative displacement component. For two-dimensional analyses the third component v_z of the vector $\mathbf{v}$ is not input; it is only input for three-dimensional analyses.

6.7.9 `.pst` File Control

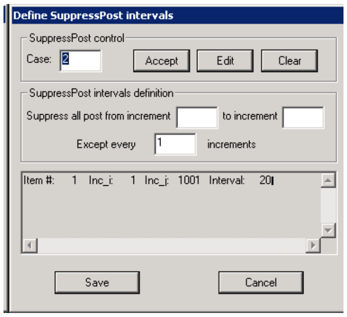
Merlin: 3.1.5

This option, Fig. 6.41 enables the user to minimize the size of the `.pst` file (used both for restart, and for the graphical postprocessor Spider. User can specify:

Case # Case number (for reference).

Load increment Specifies the range of increments inside which the .pst file is to be written at fixed interval (different from the default of 1).

Write data to .pst file every Interval of `.pst` file update.



Define SuppressPost intervals

SuppressPost control

Case: 2

SuppressPost intervals definition

Suppress all post from increment to increment

Except every 1 increments

```
Item #:    1   Inc_i:    1   Inc_j:  1001   Interval:    20
```

Figure 6.41: Suppress pst Output

6.7.10 Staged Construction/Excavation

Staged construction/excavation can be approximately simulated by the algorithms shown in Fig. 6.42 and 6.43 respectively.

Though clearly an approximation, this approach was relatively simple to implement, and yields excellent results.

1. With reference to Fig. 6.44, the user must begin by assigning the exact material properties for

   Construction all the materials, or only the reference material id. Kumo will assign an initial scaled value of E corresponding to the user entered value to the other groups.

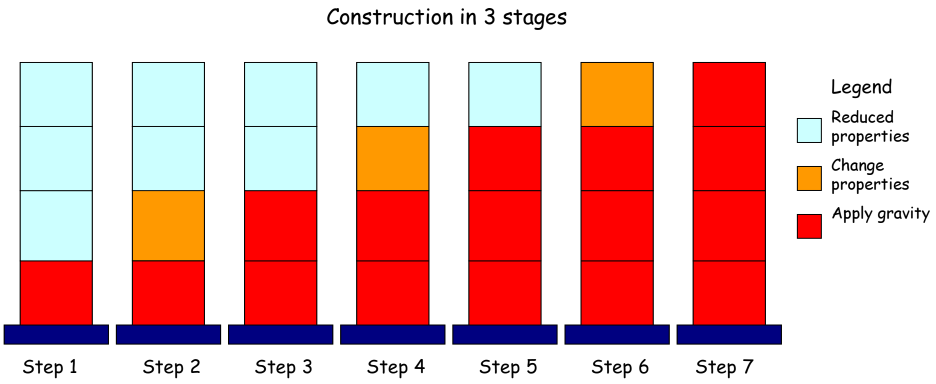
   Excavation Assign the exact material properties to all the elements.

2. Specify the reference material group id (the one with the exact properties).

3. Specify the scaling factor for E corresponding to the "ghost" elements.

4. Specify the order for construction/excavation.

It should be noted that 2 increments are needed for each additional layer of excavation or construction. Hence if not enough increments have been specified, kumo will issue a warning message, Fig. 6.45.



Step 1: define material 1 real properties (E, ν, ρ)

define "reduced" properties for all the others (E ≅ 0, ρ = 0)

apply gravity (Body Force List) only to material group 1 (all increments)

Step 2: change material 2 properties (E, ν, ρ)

Step 3 apply gravity (Body Force List) only to material groups 1 and 2 (all increments)

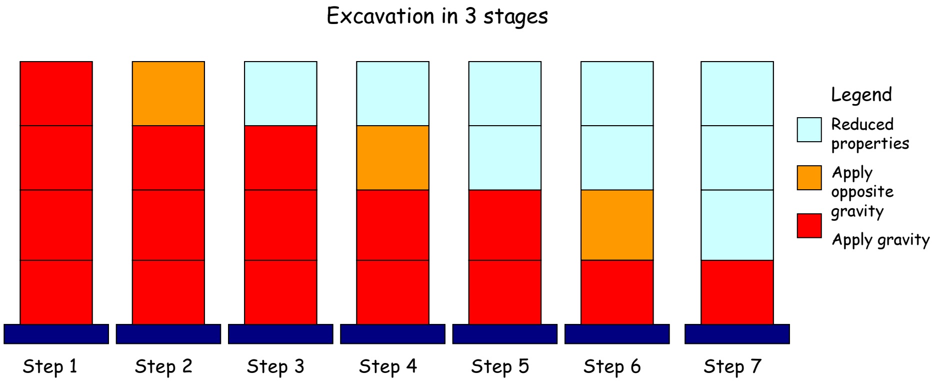
Step 4: change material 3 properties (E, ν, ρ)

Step 5: apply gravity (Body Force List) to material groups 1,2 and 3 (all increments)

Step 6: change material 4 properties (E, ν, ρ)

Step 7: apply gravity (Body Force List) to all materials (all increments)

Figure 6.42: Algorithm for Staged Construction



Step 1: define real properties (E, ν, ρ) for all materials

apply gravity (Body Force List) to all material groups (all increments)

Step 2: apply opposite gravity (Body Force List) to material groups 4 (all increments)

Step 3: change material 4 properties ("reduced" properties E, ρ ≅ 0)

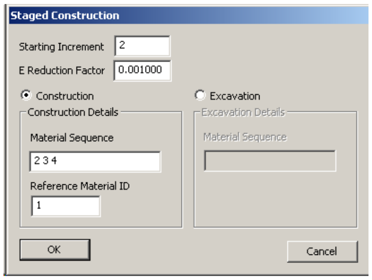
Step 4: apply opposite gravity (Body Force List) to material groups 3 (all increments)

Step 5: change material 3 properties ("reduced" properties E, ρ ≅ 0)

Step 6: apply opposite gravity (Body Force List) to material groups 2 (all increments)

Step 7: change material 2 properties ("reduced" properties E, ρ ≅ 0)

Figure 6.43: Algorithm for Staged Excavation



Starting Increment: 2

E Reduction Factor: 0.001000

● Construction ○ Excavation

Construction Details

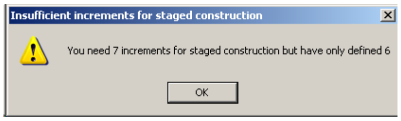
Material Sequence: 2 3 4

Reference Material ID: 1

Excavation Details

Material Sequence:

Figure 6.44: Graphical User Interface for Staged Construction/Excavation

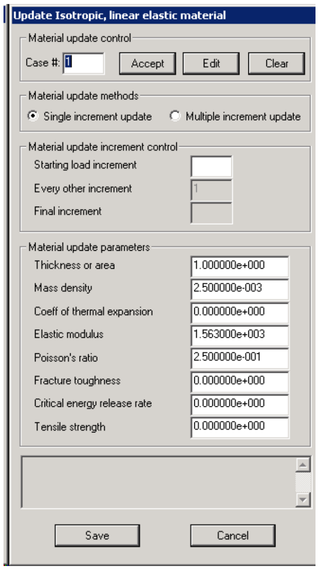


You need 7 increments for staged construction but have only defined 6

Figure 6.45: Staged COnstruction/Excavation Warning if Insufficient number of Increments Specifiedn

6.8 Incremental Material Update

This option enables the user to modify material properties after the first increment. Hence, during a load increment, user can alter any of the existing material properties, Fig. 6.46.



Material update control

Case #: 1

Material update methods

● Single increment update ○ Multiple increment update

Material update increment control

Starting load increment	
Every other increment	1
Final increment	

Material update parameters

Thickness or area	1.000000e+000
Mass density	2.500000e-003
Coeff of thermal expansion	0.000000e+000
Elastic modulus	1.563000e+003
Poisson's ratio	2.500000e-001
Fracture toughness	0.000000e+000
Critical energy release rate	0.000000e+000
Tensile strength	0.000000e+000

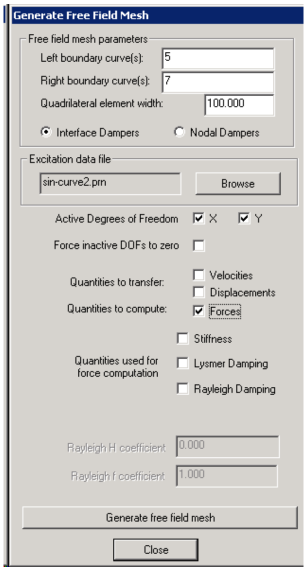
Figure 6.46: Incremental material Update

This option is useful to alter the material properties from the static ones, to dynamic ones after a restart. This can also be used to simulate stage construction by adding the mass and increasing the Young's modulus incrementally.

6.9 Generate Free Field

Fig. 6.47 shows how to generate the free field meshes.

Merlin-T: 5.3.2



Generate Free Field Mesh

Free field mesh parameters

Left boundary curve(s): 5

Right boundary curve(s): 7

Quadrilateral element width: 100.000

● Interface Dampers ○ Nodal Dampers

Excitation data file

sin-curve2.prn

Active Degrees of Freedom ☑ X ☑ Y

Force inactive DOFs to zero □

Quantities to transfer: □ Velocities □ Displacements

Quantities to compute: ☑ Forces

Quantities used for force computation □ Stiffness □ Lysmer Damping □ Rayleigh Damping

Rayleigh H coefficient 0.000

Rayleigh f coefficient 1.000

Figure 6.47: Generate Free Field

The user has the following options:

1. Specify the left and right curves (2D) or left, right, front and back patches (3D) with respect to which the free field mesh will be generated.
2. Specify type of dampers
   (a) Interface (strongly recommended) continuous element dampers
   (b) nodal dampers
3. Excitation data file (should be same as the one applied to the foundation).
4. Active degrees of freedom. If only x excitation is specified in the excitation data file, then check the x box, same with the others.
5. Force Inactive DOF to zero (recommended).
6. There are two options to transfer results of free field

   Nodal Velocities/Displacements This option is discouraged though in 2D transfer of velocities has been shown to yield good results.

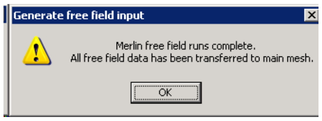
   Forces (Recommended). Results of free field analysis can be transferred as nodal forces based on

   (a) Stiffness (this will cause vertical forces)

   (b) Damping
      - Lysmer Damping (Strongly Recommended).
      - Rayleigh damping (if forces based on stiffness are being transferred). In this case user must specify the damping factor h_d corresponding to the frequency f.

6.10 Run Free Field

This will simply run all the free field analysis, and then transfer the appropriate results to the merlin input data file of the main structure, Fig. 6.48.



Merlin free field runs complete.
All free field data has been transferred to main mesh.

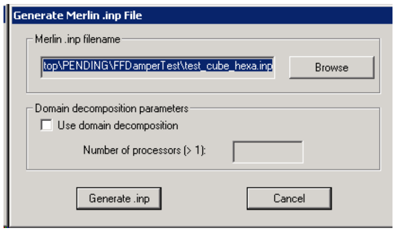
Figure 6.48: Complete Free Field Analysis Message

6.11 Write .ctrl file

A `.ctrl` file is needed to run the T3D2Merlin module. It should be noted that the Merlin input data file will be assigned the same filename as the one of the `.ctrl` file.

6.12 Generate Merlin .inp file

With all preliminary steps completed, the user may generate the *.inp-file for Merlin's analysis. The *.inp-file weds the geometric data (*.t3d-file) to the material properties and load data (*.ctrl-file). Fig. 6.47... Note that if a parallel explicit analysis is to be performed, user should specify the number of processor, and an internal domain decomposition will be performed.



Merlin .inp filename: top\PENDING\FFDamperTest\test_cube_hexa.inp

Domain decomposition parameters

□ Use domain decomposition

Number of processors (> 1):

Figure 6.49: Generate Merlin Input File

Chapter 7

Merlin Files

MerlinToolBox: Windows based driver for the following three programs.

KumoNoSu: Mesh Generator.

Merlin: Finite Element analysis program.

Spider: Graphical postprocessor

MATLAB: Based Drivers for Parameter Identification through Least Square Minimization

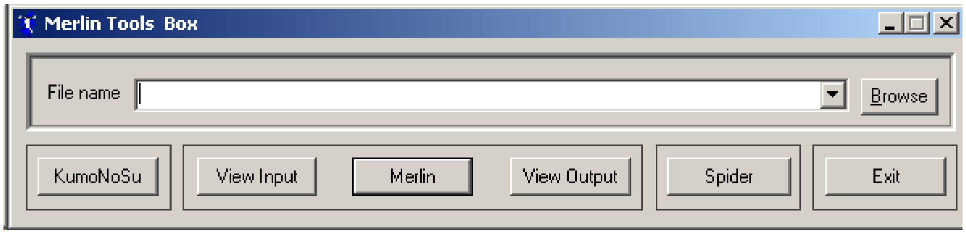


Figure 7.1: Merlin Toolbox

Extension	Used by	Generated by
Merlin Specific Files		
`.inp`	Merlin input file	KumoNoSu/T3D2Merlin
`.out`, `.dyn`	Merlin output text files	Merlin
`.pst`, `.eig`, `.rtv`	Spider input file	Merlin
KumoNoSu Specific Files		
`.bd`	T3D input file	KumoNoSu
`.t3d`	T3D output file	T3D
`.ctrl`	T3D2Merlin input file	KumoNoSu
Beaver Specific Files		
`.dam`	Beaver	Beaver
`.bd`	KumoNoSu	Beaver
`.ctrl`	KumoNoSu	Beaver

Table 7.1: File Types

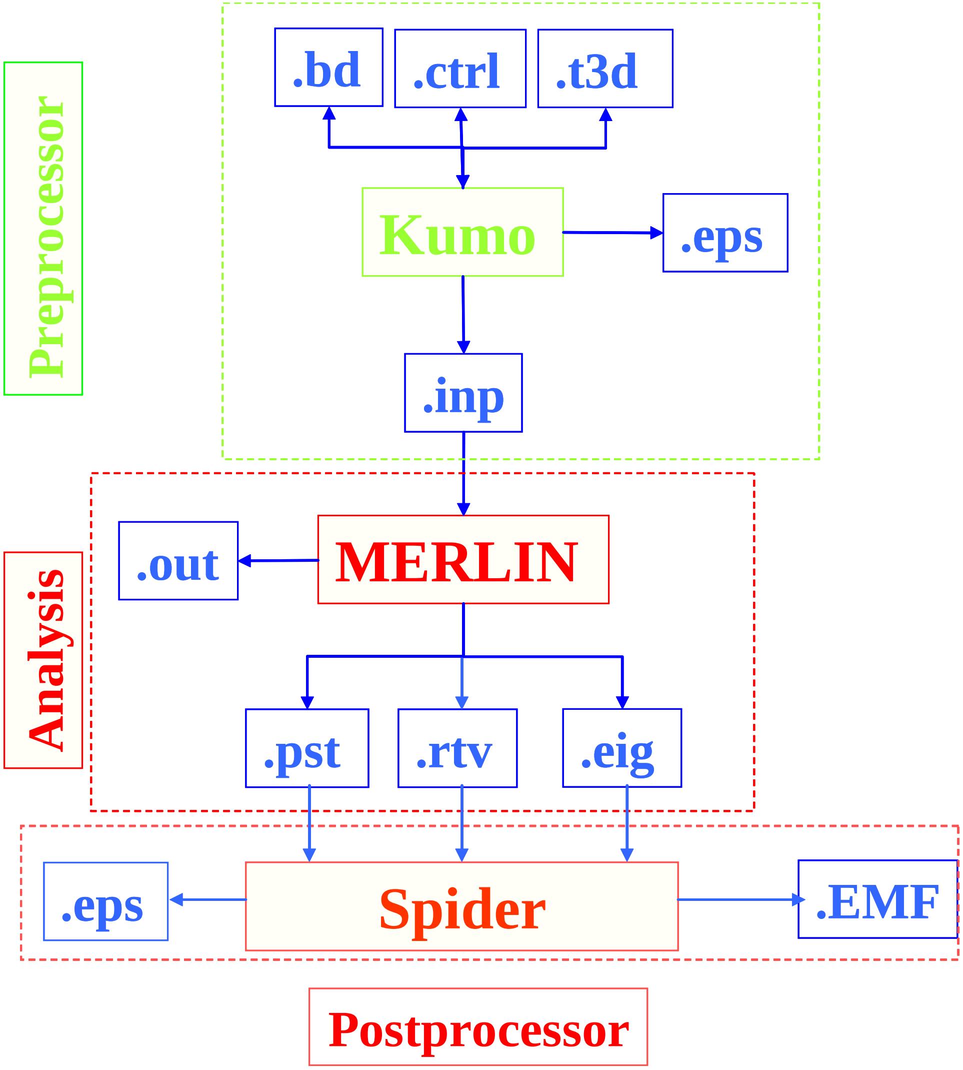


Figure 7.2: Program Interactions

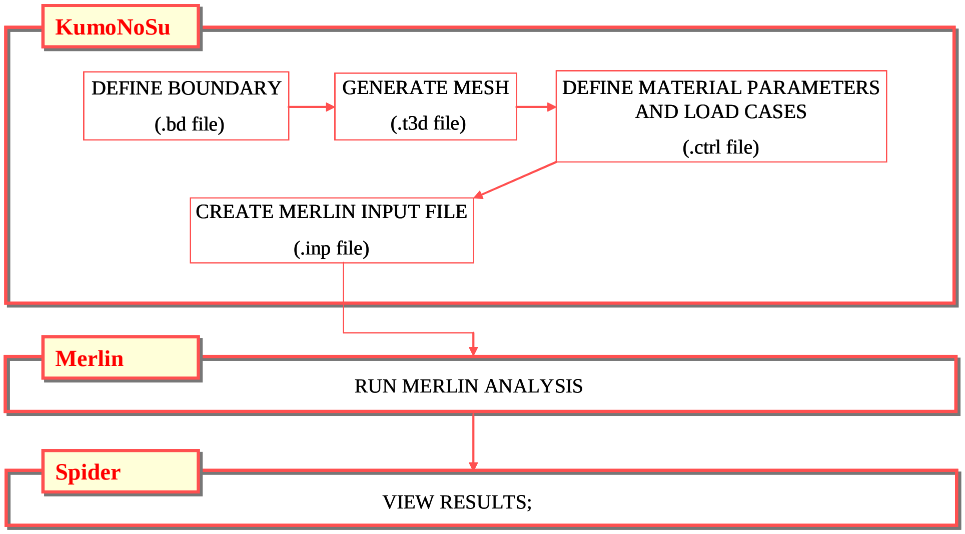


Figure 7.3: Program Interactions

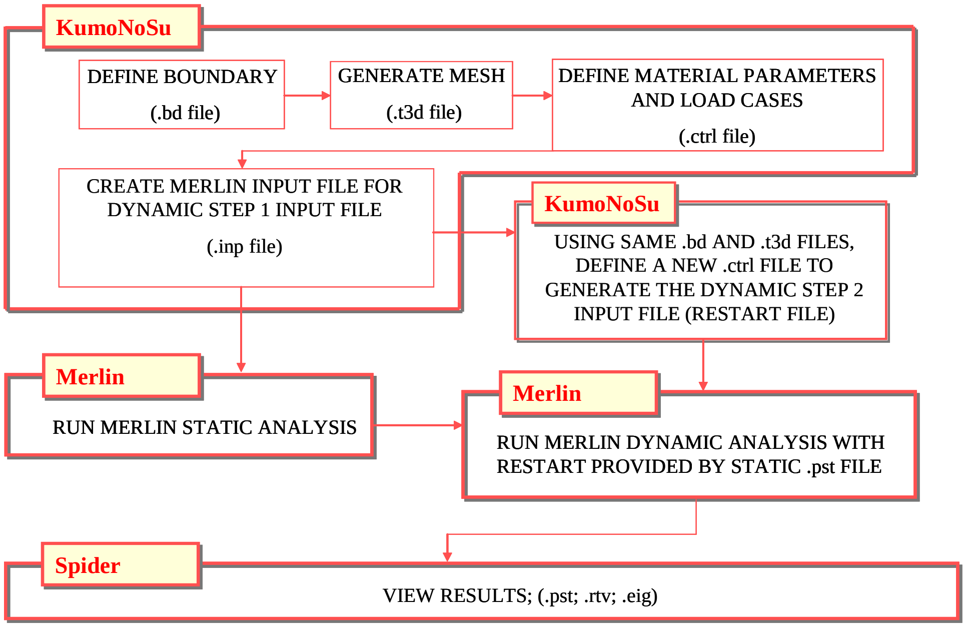

KumoNoSu
DEFINE BOUNDARY
(.bd file)
GENERATE MESH
(.t3d file)
DEFINE MATERIAL PARAMETERS
AND LOAD CASES
(.ctrl file)
CREATE MERLIN INPUT FILE FOR
DYNAMIC STEP 1 INPUT FILE
(.inp file)
KumoNoSu
USING SAME .bd AND .t3d FILES,
DEFINE A NEW .ctrl FILE TO
GENERATE THE DYNAMIC STEP 2
INPUT FILE (RESTART FILE)
Merlin
RUN MERLIN STATIC ANALYSIS
Merlin
RUN MERLIN DYNAMIC ANALYSIS WITH
RESTART PROVIDED BY STATIC .pst FILE
Spider
VIEW RESULTS; (.pst; .rtv; .eig)

Figure 7.4: Program Interactions

Entity	**Geometry**	**Topology**
Region	free	-List of boundary surfaces, patches, and shells -List of fixed vertices, curves, surfaces, patches, and shells
Surface	Rational Bezier surface	4 bounding curves -List of fixed vertices, curves, and surfaces -Parent region or surface -Regions on both sides
Patch	Plane	-List of boundary curves -List of fixed vertices and curves -Parent region -Regions on both sides
Shell	Rational Bezier surface	-List of boundary curves -List of fixed vertices and curves -Parent region -Regions on both sides
Curve	Rational Bezier curve	-2 bounding vertices -List of fixed vertices and curves -Parent region, surface, patch, shell, or curve -List of connected surfaces, patches, and shells
Vertex	Point	-Parent region, surface, patch, shell, or curve -Parent vertex -List of fixed vertices -List of connected curves

Table 7.2: Hierarchy of Model Represenatation

Appendix A

MESH GENERATION

A.1 Introduction

Finite element mesh generation is now an integral part of a finite element analysis. With the increased computational capabilities, increasingly more complex structures are being analysed. Those structures must be discretized.

The task is one of developing a mathematical model (discretization or tessalation) of a continuum model. This is not only necessary in finite elment analysis, but in computer graphics/rendering also. In computer graphics, we focus on the boundary representation, and assign colors and shades on the basis of light source and outward normal direction of the polygon.

Hence, in the most general case, meshing can be defined as the process of breaking up a physical domain into smaller sub-domains (elements) in order to facilitate the numerical solution of a partial differential equation. Surface domains may be subdivided into triangle or quadrilateral shapes, while volumes may be subdivided primarily into tetrahedra or hexahedra shapes. The shape and distribution of the elements is ideally defined by automatic meshing algorithms.

1. Point placement, followed by triangularization (discussed below).
2. Sub-domain removal. Elements are gradually removed from the domain, one ata time, until the whole domain is decomposed int finite elements.
3. Recursive subdivision. The domain is broken into simpler parts until the individual parts are single elements or simple regions, that can be meshed directly, for instance by the conformal mapping algorithm.
4. Hierarchical decomposition. The basic principle of a quadtree (or hierarchical decomposition) is to cover a planar region of interest by a square, then recursively partition squares into smaller squares until each square contains a suitably uniform subset of the input.

A.2 Triangulation

The concept of Voronoi diagrams first appeared in works of Descartes as early as 1644. Descartes used Voronoi-like diagrams to show the disposition of matter in the solar system and its environs.

The first man who studied the Voronoi diagram as a concept was a German mathematician G. L. Dirichlet. He studied the two- and three dimensional case and that is why this concept is also known as Dirichlet tessellation. However it is much better known as a Voronoi diagram because another German mathematician M. G. Voronoi in 1908 studied the concept and defined it for a more general n-dimensional case.

Very soon after it was defined by Voronoi it was developed independently in other areas like meteorology and crystalography. Thiessen developed it in meteorology in 1911 as an aid to computing more accurate estimates of regional rainfall averages. In the field of crystalography German researchers dominated and Niggli in 1927 introduced the term Wirkungsbereich (area of influence) as a reference to a Voronoi diagram.

During the years this concept kept being rediscovered over and over again in different fields of science and today it is extensively used in about 15 different fields of sciences. Some of them being mathematics, computer science, biology, cartography, physiology and many others.

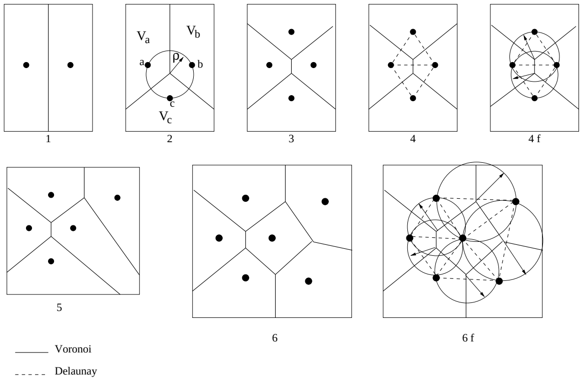


Figure A.1: Voronoi and Delaunay Tessellation

A.2.1 Voronoi Polygon

Given a finite set of poits in the plane, the idea is to assign to each point a region of influence in such a way that the regions decompose the plane, (**?**). To describe a specific way to do that, let S be a subset of $\mathbb{R}^2$ ($S \subseteq \mathbb{R}^2$). We define the **Voronoi region** of $p \in S$ as the set of points $x \in \mathbb{R}$ that are at least as close to p as to any other points in S:

$$V_p = \big\{x \in \mathbb{R}^2 \mid ||x-p|| \leq \| x-1 \|, \forall q \in S\big\} \tag{A.1}$$

Each point $x \in \mathbb{R}^2$ has at least one nearest point in S, so it lies in at least one Voronoi region. Two Voronoi regions lie on opposite sides of the perpendicular bisector separating the two generating points.

A.2.2 Delaunay Triangulation

The dual of the Voronoi diagram is obtained by drawing straight *Delaunay edges* connecting points $p, q \in S$ if and only if their Voronoi regions intersect along a common line segment. Thus in general, the Delaunay edges decompose the convex hull of S into triangular regions which are referred to as **Delaunay triangles**, (**?**).

Using Euler's relation, it can be shown that a planar graph with $n \geq 3$ vertices has at most $3n-6$ edges and at most $2n-4$ faces. THe same bounds hold for the number of Delaunay edges and triangles. Each Voronoi vertex $u = V_a \cap V_b \cap V_c$ is the center of a circle with radius $\rho = \| u-a \| = \| u-b \| = \| u-c \|$. The circle is **empty** because it encloses no point of S.

Additional detailed information on Voronoi tesselation, (**?**) is an excellent reference.

A.2.3 MATLAB Code

```
rand('state',4);
x = rand(1,3);
y = rand(1,3);
TRI = delaunay(x,y);
subplot(1,2,1),...
trimesh(TRI,x,y,zeros(size(x))); view(2),...
```

```
axis([0 1 0 1]); hold on;
plot(x,y,'o');
set(gca,'box','on');

[vx, vy] = voronoi(x,y,TRI);
subplot(1,2,2),...
plot(x,y,'r+',vx,vy,'b-'),...
axis([0 1 0 1])
```

A.3 Finite Element Mesh Generation

A.3.1 Boundary Definition

In order to discretize the continuum into a finite element mesh, first key geometrical information of the must be specified hierarchically:

Vertices: with nodal coordinates, and approximate desired element size in the immediate vicinity (thus describing the mesh density).

Edges: which connect vertices. Those can be either linear segments, polylines, or curves.

Surfaces: composed of edges, defined counterclockwise.

Volumes: (3D only) composed of surfaces.

Associated with surfaces (2D0, or volumes (3D) are different material properties.

Examples of the hierarchical boundary definition is shown in Fig. A.2 and A.3.

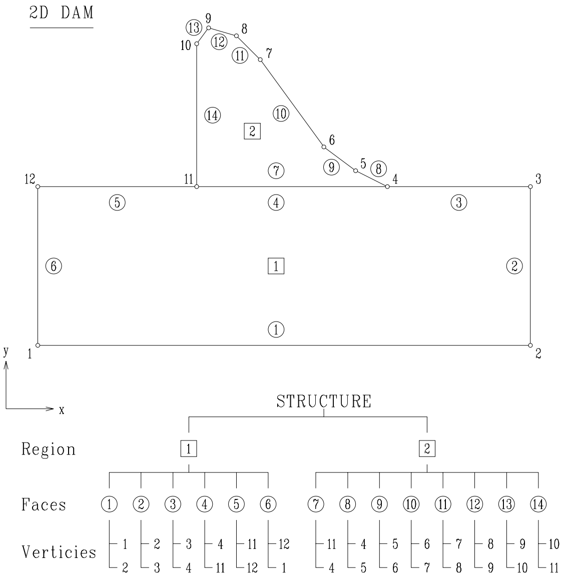


Figure A.2: Control Point for a 2D Mesh

A.3.2 Interior Node Generation

Once the boundary has been defined, we need to insert internal nodes at a spacing which respect the required mesh density. There are numerous techniques to insert those internal nodes. We present one approach, FIg. A.4

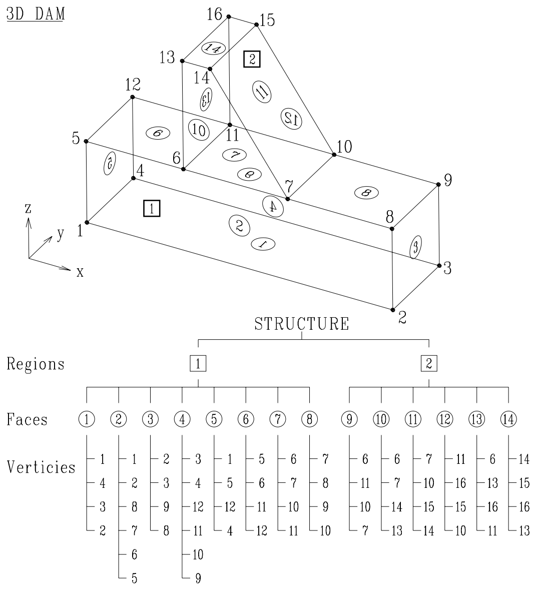


Figure A.3: Control Point for a 3D Mesh

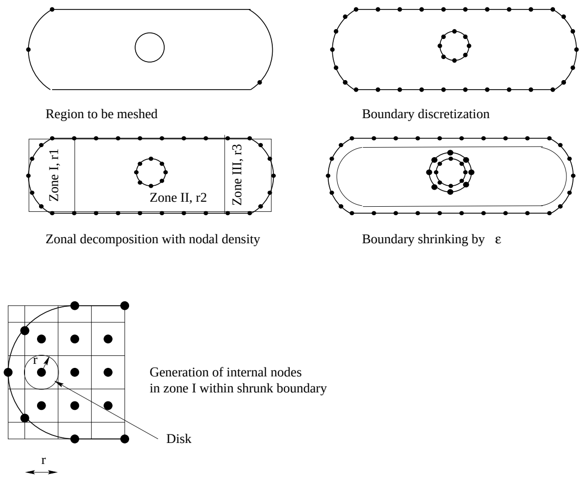


Figure A.4: A Two Dimensional Triangularization AlgorithmControl Point for a 3D Mesh

1. Decompose the region into a disjoint ensemble of subregions with equal mesh density.

2. Shrink the mesh to avoid elements near the boundary with very acute interior angles.

3. Starting with the first zone, circumscribe it by the smallest possible rectangle.

4. Superimpose a square rectangular grid over the circumscribing rectangle.

5. Use a random number generator to randomly generate one interior node in each square. A disk of radius r centered at each node is used to test that no other surrounding nodes are enclosed in the disk. If so, the node in question is regenerated.

An alternative approach consists in, (Červenka, J. 1994)

1. Generating a triangularization compatible with the initial nodes.

2. Check lengths of the edges. If an edge does not satisfy the prescribed size r, a new node is inserted in the center of the edge. The prescribed size is interpolated between those of the vertices at each end of the edge.

3. Repeat this operation until convergence.

4. Smoothen the elemnts to assure appropriate aspect ratios.

A.3.3 Final Triangularization

With boundary and interior nodes generated:

1. Determine the Voronoi polygons

2. Perform a Delaunay triangularization

3. Smoothen the mesh to ensure that all generated elements have a satisfactory aspect ratio.

It should be noted that recent algorithms, which can generate quadrilateral elements out of the Delaunay triangularization have recently emerged.

Appendix B

Rational Bezier Curve

The rational Bezier curve is determined by a control polygon, as shown in Fig. B.1.

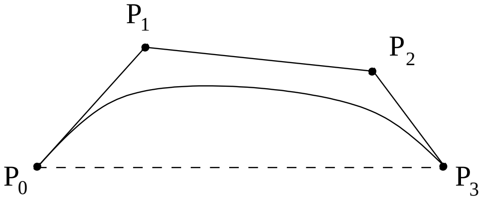


Figure B.1: Example of Rational Bezier Curve and Its Control Polygon

The curve generally follows the shape of the control polygon, the first and the last points on the curve are coincident with the first P_0 and the last point P_3 of the control polygon, and the the first and the last segments of the control polygon coincide with the curve tangent at the starting and the ending points of the curve, respectively.

Mathematically, it can be represented by the following equation(**?**).

$$P(t) = \frac{\sum_{i=0}^{n} w_i \, P_i \, B_i^n(t)}{\sum_{i=0}^{n} w_i \, B_i^n(t)}, \tag{B.1}$$

where $P(t)$ is the point on the curve, P_i are Bezier control points, w_i are weight of Bezier control points, B_i^n is Bernstein polynomials, t is an independent variable varying in range from 0 to 1, and n is the degree of the polynomial defining the curve segment.

Bernstein polynomial function is defined as

$$B_i^n(t) = \left(\begin{array}{c} n \\ i \end{array} \right) t^i (1-t)^{n-i} \tag{B.2}$$

or recursively as

$$B_i^n(t) = (1-t)\, B_i^{n-1}(t) + t\, B_{i-1}^{n-1}(t), \tag{B.3}$$

where $B_0^0 = 1$.

The curve order is equal to the number of polygon points $n+1$. Therefore,a 4 point rational bezier polygons results in a cubic curve whereas a 3 point one results in a quadratic curve.

In the mesh generator program, the point P_0 and P_n correspond to the model vertices while the remaining points form the control polygon of the curve. In the particle model, the elliptical curve has been used to define the inclusion. Therefore, to construct the rational Bezier curve the control points of the elliptical curve must be determined based on the degree of the polynomial of the elliptical curve.

B.1 Quadratic curve

In the *T3D* (**?**) mesh generator used internally by *Kumo*(**?**), the quadratic curve of the elliptical arc is defined as demonstrated in Fig. B.2.

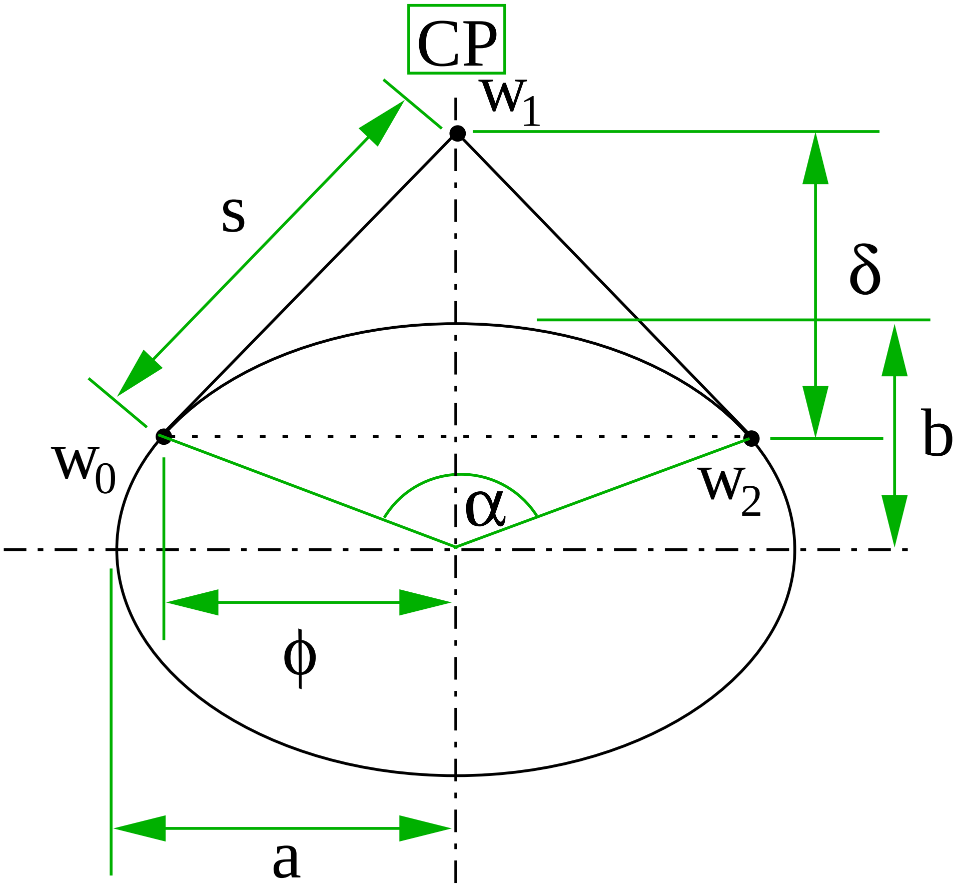


Figure B.2: Elliptical Arc-Quadratic Curve

Equations for each the variables shown in Fig. B.2 can be summarized as follows.

$$\delta = b\frac{\sin^2 \frac{\alpha}{2}}{\cos \frac{\alpha}{2}} \tag{B.4}$$

$$\phi = a \sin \frac{\alpha}{2} \tag{B.5}$$

$$s = \frac{\sin \frac{\alpha}{2}}{\cos \frac{\alpha}{2}} \sqrt{a^2 \cos^2 \frac{\alpha}{2} + b^2 \sin^2 \frac{\alpha}{2}} \tag{B.6}$$

$$w_0 = w_2 = 1 \tag{B.7}$$

$$w_1 = \cos \frac{\alpha}{2} \tag{B.8}$$

$$\alpha \in (0 : 2\pi) \backslash \{\pi\} \tag{B.9}$$

where w_0, w_1, and w_2 are weights of the bezier control points.

The y coordinate of the control point CP is then determined by

$$y_{CP} = \frac{\phi}{\tan \frac{\alpha}{2}} + \delta \tag{B.10}$$

In the general case where the starting and the ending points are not in the symmetrical pattern shown in Fig. B.2, Fig.B.3, determination of the control point and its weight requireds special attaintion.

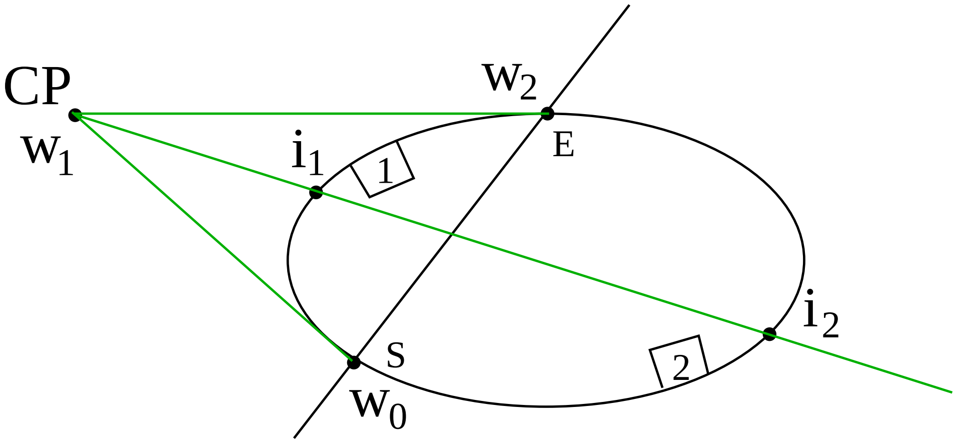


Figure B.3: General Case of Elliptical Arc-Quadratic Curve

Assuming that S is the starting point of the curve 1 (or 2) and E is the ending point of the curve 1 (or 2). To determine the control point of both curve 1 and 2, first the tangent to the elliptical curve at the starting S and the ending E point are drawn and the intersection of the 2 tangent lines is the control point CP. It was shown by (**?**) that the weight w_0 and w_2 of both curve 1 and 2 are the same and are equal to 1. The weight w_1 for both curve can be determined by following equations.

1. Curve 1

$$w_1 = \frac{M - I_1}{I_1 - CP} \tag{B.11}$$

2. Curve 2

$$w_2 = \frac{M - I_2}{I_2 - CP} \tag{B.12}$$

where M is the coordinate of the mid point of the line SE, I_1 is the coordinate of the intersection point between the elliptical curve 1 and the line originating from the control point passing through point M and I_2 is the coordinate of the intersection point between the elliptical curve 2 and the line originating from the control point passing through point M.

B.2 Cubic curve

In the $T3D$ mesh generator (**?**), the cubic curve of the elliptical arc is defined in Fig. B.4.

All equations of each variables shown in Fig. B.4 are summarized as follows.

$$\delta = 2b\frac{\sin^2\frac{\alpha}{2}}{1+2\cos\frac{\alpha}{2}} \tag{B.13}$$

$$\phi = a\frac{\sin\alpha}{1+2\cos\frac{\alpha}{2}} \tag{B.14}$$

$$s = \frac{2\sin\frac{\alpha}{2}}{1+2\cos\frac{\alpha}{2}}\sqrt{a^2\,\cos^2\frac{\alpha}{2}+b^2\,\sin^2\frac{\alpha}{2}} \tag{B.15}$$

$$w_0 = w_3 = 1 \tag{B.16}$$

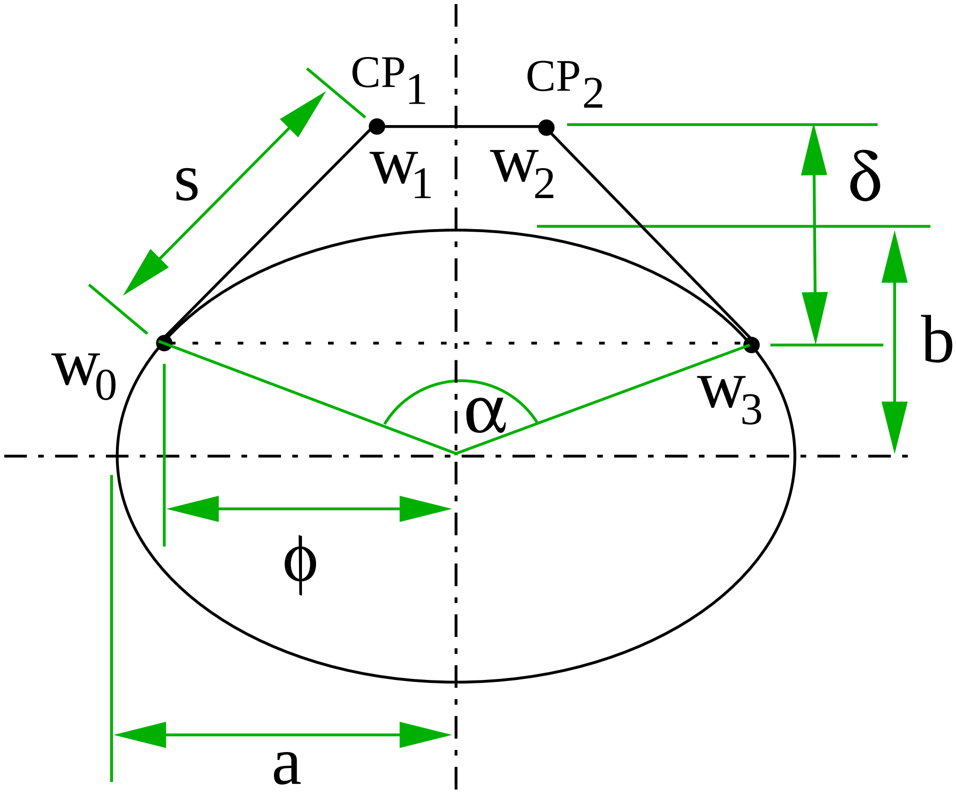


Figure B.4: Elliptical Arc-Cubic Curve

$$w_1 = w_2 = \frac{1 + 2\cos\frac{\alpha}{2}}{3} \tag{B.17}$$

$$\alpha \in (0 : 2\pi)\backslash\left\{\frac{4}{3}\pi\right\} \tag{B.18}$$

where w_0, w_1, w_2 and w_3 are weights of the Bezier control points.

The control points CP_1 and CP_2 are the points which have the distance s from starting point SP and the ending point EP in the direction of the tangent line to the curve at the starting and the ending points, respectively.

In the case where the starting and the ending points are not in the symmetrical pattern as shown in Fig. B.5, and both points are opposite to each other as shown in Fig.B.5.

The value of the weights of the Bezier curve are still the same as shown in Eq. B.16 and B.17. However, the coordinates of both control points CP_1 and CP_2 can be computed as follows.

From Fig.B.5, suppose SP and EP are the starting and the ending points of curve 1.

1. Draw the tangent line to the elliptical curve at the SP and EP and determine the slope of the tangent s_1.

2. Draw a line from the origin O, which a slope equal to s_1. This line intersects with curve 1 at point P.

3. Draw a tangent line to the elliptical curve 1 at point P (slope = s_2) until it intersects the tangent line originating from the staring and the ending points at point A and B, respectively.

4. Determine the distance d.

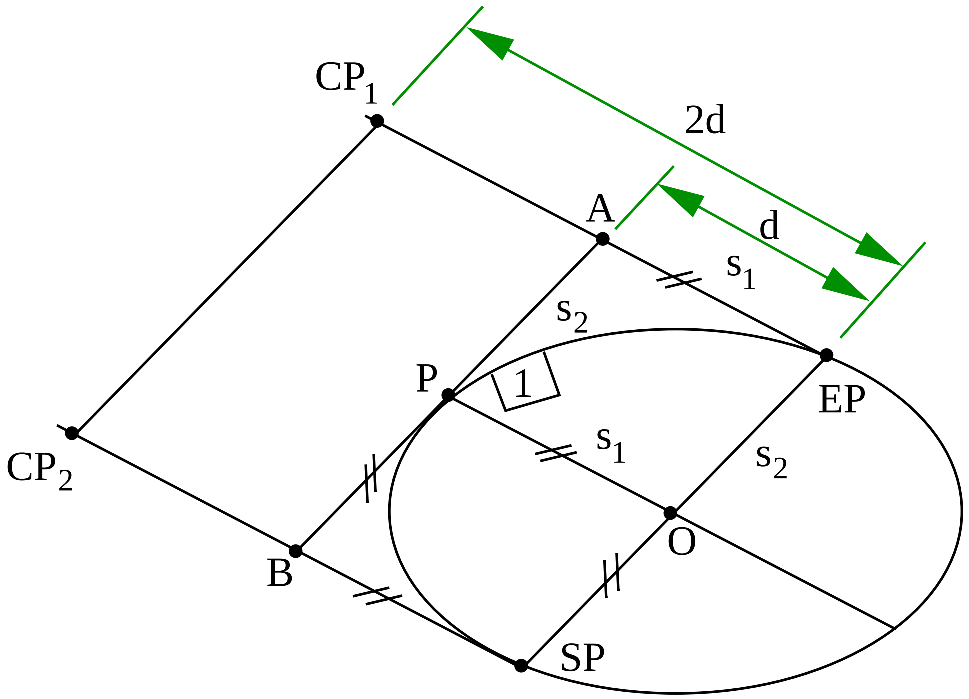


Figure B.5: Specific Case of Elliptical Arc-Cubic Curve

5. Control points CP_1 and CP_2 are then the points from the starting and the ending points with distance $2d$ in the direction of its tangent, respectively.

Appendix C

Examples of Finite Element Boundary Definition Output Files from PARSIFAL Program

C.1 Matrix and Inclusion without Interface elements

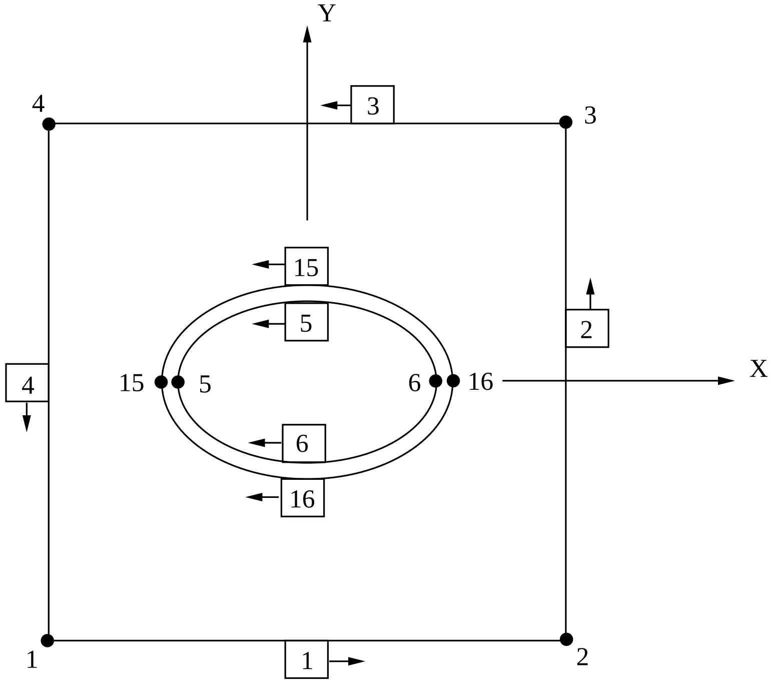


Figure C.1: Matrix and inclusion without interface elements

Fig.C.1 illustrates an elliptical inclusion (inside ellipse) surrounded by a matrix (box), with a perfectly rigid interface (outside ellipse) between the matrix and inclusion. The boundary description for this problem is defined using (8) vertices, (8) curves, (3) patches, and (4) master/slave relations. The resulting .bd file can be shown as follows.

```
vertex 1  xyz -1.00000000e+000  -1.00000000e+000  0.00000000e+000
vertex 2  xyz 1.00000000e+000  -1.00000000e+000  0.00000000e+000
vertex 3  xyz 1.00000000e+000  1.00000000e+000  0.00000000e+000
vertex 4  xyz -1.00000000e+000  1.00000000e+000  0.00000000e+000
vertex 5  xyz -5.00000000e-001  0.00000000e+000  0.00000000e+000
vertex 6  xyz 5.00000000e-001  0.00000000e+000  0.00000000e+000
vertex 15  xyz -5.00000000e-001  0.00000000e+000  0.00000000e+000
vertex 16  xyz 5.00000000e-001  0.00000000e+000  0.00000000e+000

curve 1  order 2  vertex 1  2
curve 2  order 2  vertex 2  3
curve 3  order 2  vertex 3  4
curve 4  order 2  vertex 4  1
curve 5  order 4  vertex 6  5
polygon 1 xyz 5.00000000e-001  2.50000000e-001  0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001  2.50000000e-001  0.00000000e+000
weight 3.333e-001
curve 6  order 4  vertex 6  5
polygon 1 xyz 5.00000000e-001  -2.50000000e-001  0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001  -2.50000000e-001  0.00000000e+000
weight 3.333e-001
curve 15  order 4  vertex 16  15
polygon 1 xyz 5.00000000e-001  2.50000000e-001  0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001  2.50000000e-001  0.00000000e+000
weight 3.333e-001
curve 16  order 4  vertex 16  15
polygon 1 xyz 5.00000000e-001  -2.50000000e-001  0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001  -2.50000000e-001  0.00000000e+000
weight 3.333e-001

patch 1 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary curve 5 -6
size 5.000e-002  property 1
patch 2 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary curve 15 -16
size def hole
patch 3 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary curve 1 2 3 4
size 1.000e-001 property 2 subpatch 2

masterslave 1 vertex 5 15
masterslave 2 vertex 6 16
masterslave 3 curve 5 15
masterslave 4 curve 6 16
```

Note that vertices 5 and 15 share the same coordinates, as do vertices 6 and 16. Additionally, curves 5/15 and 6/16 share identical properties. However, these curves combine to form separate patches, with curves 5,6 forming patch 1 and curves 15,16 forming patch 2. Additionally, patch 2 is defined as a hole, since it is simply providing the connection between the inclusion (patch 1) and the matrix (patch 3). Patch 2 serves as a subpatch for patch 3.

In order to enforce the rigid connection between the matrix and the inclusion, master/slave relationships are defined for vertices 5/15 and 6/16, and for curves 5/15 and 6/16. These relations effectively join the matrix and inclusion in the resulting MERLIN .inp file.

C.2 Matrix and Inclusion with Interface Elements

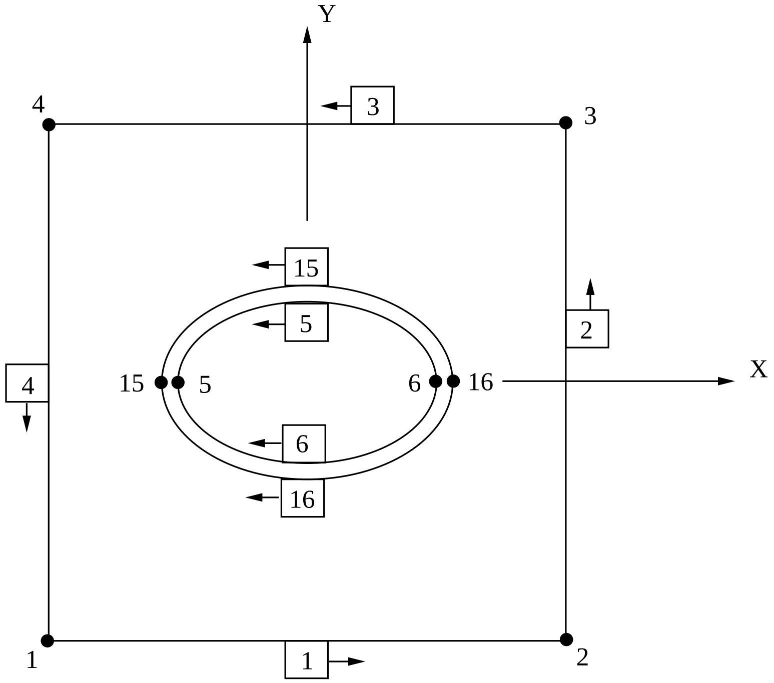


Figure C.2: Matrix and inclusion with interface elements

Fig.C.2 illustrates an elliptical inclusion (inside ellipse) surrounded by a matrix (box), with interface elements (outside ellipse) between the matrix and inclusion. The boundary description for this problem is defined using (8) vertices, (8) curves, (3) patches, (2) crack segments, and (2) crack paths. The resulting .bd file can be shown as follows.

```
# This input file generated by T3d preprocessor at 15:56:28 on 11/05/01

vertex 1  xyz -1.00000000e+000  -1.00000000e+000  0.00000000e+000
vertex 2  xyz 1.00000000e+000  -1.00000000e+000  0.00000000e+000
vertex 3  xyz 1.00000000e+000  1.00000000e+000  0.00000000e+000
vertex 4  xyz -1.00000000e+000  1.00000000e+000  0.00000000e+000
vertex 5  xyz -5.00000000e-001  0.00000000e+000  0.00000000e+000
vertex 6  xyz 5.00000000e-001  0.00000000e+000  0.00000000e+000
vertex 15  xyz -5.00000000e-001  0.00000000e+000  0.00000000e+000
vertex 16  xyz 5.00000000e-001  0.00000000e+000  0.00000000e+000

curve 1  order 2  vertex 1  2
curve 2  order 2  vertex 2  3
curve 3  order 2  vertex 3  4
curve 4  order 2  vertex 4  1
curve 5  order 4  vertex 6  5
polygon 1 xyz 5.00000000e-001  2.50000000e-001  0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001  2.50000000e-001  0.00000000e+000
```

```
weight 3.333e-001
curve 6  order 4  vertex 6  5
polygon 1 xyz 5.00000000e-001  -2.50000000e-001  0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001  -2.50000000e-001  0.00000000e+000
weight 3.333e-001
curve 15  order 4  vertex 16  15
polygon 1 xyz 5.00000000e-001  2.50000000e-001  0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001  2.50000000e-001  0.00000000e+000
weight 3.333e-001
curve 16  order 4  vertex 16  15
polygon 1 xyz 5.00000000e-001  -2.50000000e-001  0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001  -2.50000000e-001  0.00000000e+000
weight 3.333e-001

patch 1 normal 0.0000e+000 0.0000e+000 1.0000e+000  boundary curve 5 -6
size 5.000e-002 property 1
patch 2 normal 0.0000e+000 0.0000e+000 1.0000e+000  boundary curve 15 -16
size def hole
patch 3 normal 0.0000e+000 0.0000e+000 1.0000e+000  boundary curve 1 2 3 4
size 1.000e-001 property 2 subpatch 2

crack NLFM
crack 1 structure curve 15 5
crack 2 structure curve 16 6
crack structure path 1 property 3
crack structure path 2 property 3
```

Note that vertices 5 and 15 share the same coordinates, as do vertices 6 and 16. Additionally, curves 5/15 and 6/16 share identical properties. However, these curves combine to form separate patches, with curves 5,6 forming patch 1 and curves 15,16 forming patch 2. Additionally, patch 2 is defined as a hole, since it is simply providing the connection between the inclusion (patch 1) and the matrix (patch 3). Patch 2 serves as a subpatch for patch 3.

In order to insert the interface elements between the matrix and the inclusion, structure crack segments are defined between curves 5/15 and 6/16, and a two crack paths are defined using these two crack segments. Since the crack type is defined as NLFM, interface elements will be inserted between the matrix and inclusion in the MERLIN .inp file.

C.3 Matrix,Inclusion, and propagating Crack with Interface Elements

Fig.C.3 illustrates an elliptical inclusion (inside ellipse) surrounded by a matrix (box), with interface elements (outside ellipse) between the matrix and inclusion. Additionally, a crack extends from the right side of the inclusion (triangle). This crack, while represented by a triangle with a finite crack mouth opening in the above picture, is considered to have a zero thickness in the .bd file. The boundary description for this problem is defined using (10) vertices, (10) curves, (3) patches, (3) crack segments, and (2) crack paths. The resulting .bd file can be shown as follows.

```
# This input file generated by T3d preprocessor at 12:47:58 on 11/06/01

vertex 1  xyz -1.00000000e+000  -1.00000000e+000  0.00000000e+000
```

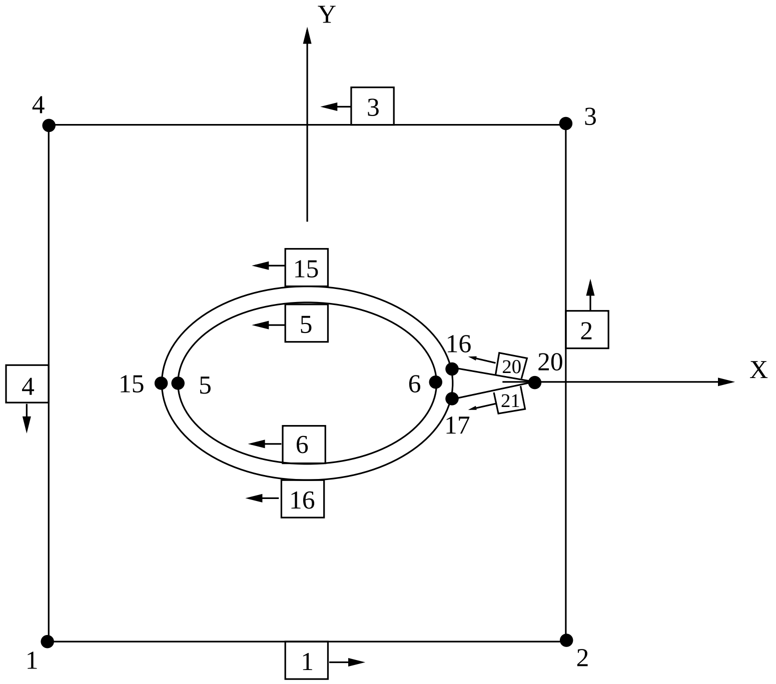


Figure C.3: Matrix, inclusion, and propagating crack with interface elements

```
vertex 2  xyz 1.00000000e+000  -1.00000000e+000  0.00000000e+000
vertex 3  xyz 1.00000000e+000  1.00000000e+000  0.00000000e+000
vertex 4  xyz -1.00000000e+000  1.00000000e+000  0.00000000e+000
vertex 5  xyz -5.00000000e-001  0.00000000e+000  0.00000000e+000
vertex 6  xyz 5.00000000e-001  0.00000000e+000  0.00000000e+000
vertex 15  xyz -5.00000000e-001  0.00000000e+000  0.00000000e+000
vertex 16  xyz 5.00000000e-001  0.00000000e+000  0.00000000e+000
vertex 17  xyz 5.00000000e-001  0.00000000e+000  0.00000000e+000
coincide vertex 16
vertex 20  xyz 7.50000000e-001  0.00000000e+000  0.00000000e+000

curve 1  order 2  vertex 1  2
curve 2  order 2  vertex 2  3
curve 3  order 2  vertex 3  4
curve 4  order 2  vertex 4  1
curve 5  order 4  vertex 6  5
polygon 1 xyz 5.00000000e-001  2.50000000e-001  0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001  2.50000000e-001  0.00000000e+000
weight 3.333e-001
curve 6  order 4  vertex 6  5
polygon 1 xyz 5.00000000e-001  -2.50000000e-001  0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001  -2.50000000e-001  0.00000000e+000
weight 3.333e-001
curve 15  order 4  vertex 16  15
polygon 1 xyz 5.00000000e-001  2.50000000e-001  0.00000000e+000
weight 3.333e-001
```

```
polygon 2 xyz -5.00000000e-001  2.50000000e-001  0.00000000e+000
weight 3.333e-001
curve 16  order 4  vertex 17  15
polygon 1 xyz 5.00000000e-001  -2.50000000e-001  0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001  -2.50000000e-001  0.00000000e+000
weight 3.333e-001
curve 20  vertex 20  16
curve 21  vertex 20  17  coincide curve 20

patch 1 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary curve 5 -6
size 5.000e-002  property 1
patch 2 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary
curve 20 15 -16 -21 size def hole
patch 4 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary curve 1 2 3 4
size 1.000e-001 property 2 subpatch 2

crack NLFM
crack 1 structure curve 21 20 front2d 20
crack 2 structure curve 5 15
crack 3 structure curve 16 6
crack structure path 1 mat 3
crack structure path 2 3 mat 3
```

Note that vertices 5 and 15 share the same coordinates, as do vertices 6, 16, and 17. Vertices 16 and 17, which form the mouth of the crack, are labeled as coincident. The crack propagating from the inclusion is formed using curves 20 and 21 (coincident), with vertex 20 serving as the crack front. Curves 5/15 and 6/16 also share identical properties. However, these curves combine to form separate patches, with curves 5,6 forming patch 1 and curves 15,16 (with curves 20 and 21) forming patch 2. Additionally, patch 2 is defined as a hole, since it is providing the interface connection between the inclusion (patch 1) and the matrix (patch 3). Patch 2 serves as a subpatch for patch 3.

Two separate crack paths are considered. The first is the crack propagating from the inclusion. This crack path is defined by a single crack segment formed by curves 20 and 21, with the crack front at vertex 20. The second crack path surrounds the inclusion, and is formed by two crack segments (curves 5/15 and 6/16).

C.4 Matrix, Inclusion, and Two Propagating Cracks with Interface Elements

Fig.C.4 illustrates an elliptical inclusion (inside ellipse) surrounded by a matrix (box), with interface elements (outside ellipse) between the matrix and inclusion. Additionally, cracks extends from the right and left sides of the inclusion (triangle). This crack, while represented by a triangle with a finite crack mouth opening in the above picture, is considered to have a zero thickness in the .bd file. The boundary description for this problem is defined using (12) vertices, (12) curves, (3) patches, (4) crack segments, and (4) crack paths. The resulting .bd file can be shown as follows.

```
# This input file generated by Kumonosu at 11:48:44 on 11/07/01

vertex 1  xyz -1.00000000e+000  -1.00000000e+000  0.00000000e+000
vertex 2  xyz 1.00000000e+000  -1.00000000e+000  0.00000000e+000
vertex 3  xyz 1.00000000e+000  1.00000000e+000  0.00000000e+000
vertex 4  xyz -1.00000000e+000  1.00000000e+000  0.00000000e+000
vertex 5  xyz -5.00000000e-001  0.00000000e+000  0.00000000e+000
vertex 6  xyz 5.00000000e-001  0.00000000e+000  0.00000000e+000
```

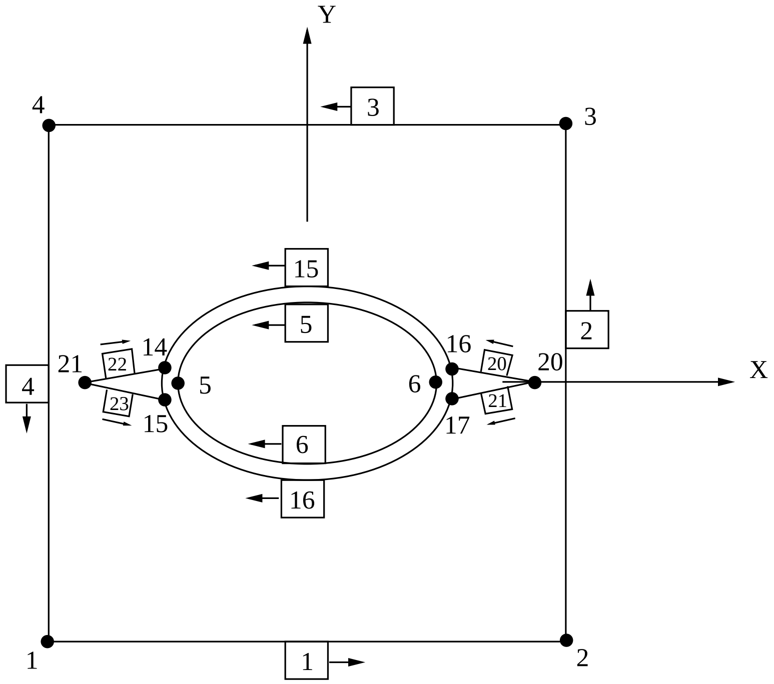


Figure C.4: Matrix, inclusion, and two propagating cracks with interface elements

```
vertex 14  xyz -5.00000000e-001  0.00000000e+000  0.00000000e+000
vertex 15  xyz -5.00000000e-001  0.00000000e+000  0.00000000e+000
coincide vertex 14
vertex 16  xyz 5.00000000e-001  0.00000000e+000  0.00000000e+000
vertex 17  xyz 5.00000000e-001  0.00000000e+000  0.00000000e+000
coincide vertex 16
vertex 20  xyz 7.50000000e-001  0.00000000e+000  0.00000000e+000
vertex 21  xyz -7.50000000e-001  0.00000000e+000  0.00000000e+000

curve 1  order 2  vertex 1  2
curve 2  order 2  vertex 2  3
curve 3  order 2  vertex 3  4
curve 4  order 2  vertex 4  1
curve 5  order 4  vertex 6  5
polygon 1 xyz 5.00000000e-001  2.50000000e-001  0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001  2.50000000e-001  0.00000000e+000
weight 3.333e-001
curve 6  order 4  vertex 6  5
polygon 1 xyz 5.00000000e-001  -2.50000000e-001  0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001  -2.50000000e-001  0.00000000e+000
weight 3.333e-001
curve 15  order 4  vertex 16  14
polygon 1 xyz 5.00000000e-001  2.50000000e-001  0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001  2.50000000e-001  0.00000000e+000
weight 3.333e-001
```

```
curve 16  order 4  vertex 17  15
polygon 1 xyz 5.00000000e-001  -2.50000000e-001  0.00000000e+000
weight 3.333e-001
polygon 2 xyz -5.00000000e-001  -2.50000000e-001  0.00000000e+000
weight 3.333e-001
curve 20  vertex 20  16
curve 21  vertex 20  17  coincide curve 20
curve 22  vertex 21  14
curve 23  vertex 21  15  coincide curve 22

patch 1 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary curve 5 -6
size 5.000e-002  property 1
patch 2 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary curve
20 15 -22 23 -16 -21   size def hole
patch 4 normal 0.0000e+000 0.0000e+000 1.0000e+000 boundary curve 1 2 3 4
size 1.000e-001 property 2 subpatch 2

crack NLFM
crack 1 structure curve 21 20 front2d 20
crack 2 structure curve 22 23 front2d 21
crack 3 structure curve 5 15
crack 4 structure curve 16 6
crack structure path 1 property 3
crack structure path 2 property 3
crack structure path 3 property 3
crack structure path 4 property 3
```

Unlike the example with a single propagating crack, in which the interface around the inclusion was treated as a single crack path, this geometry requires that (4) crack paths are defined: one for the left propagating crack, one for the right propagating crack, and two paths around the inclusion (top and bottom). The rest of the boundary description is similar to that of the single crack example. There are (3) total patches: one for the inclusion, a second composed of the inclusion and the two cracks (defined as a hole), and a third which is composed of the matrix.

C.5 Matrix and Interior Crack with Interface Elements

Fig. C.5 illustrates an interior crack (red lines) surrounded by a matrix (yellow box). The boundary description for this problem is defined using (8) vertices, (8) curves, (1) patch, (2) crack segments, and (2) crack paths. The resulting .bd file is on the next page.

```
# This input file generated by Kumonosu at 14:21:05 on 11/29/01

vertex 1  xyz -1.00000000e+000  -1.00000000e+000  0.00000000e+000
vertex 2  xyz 1.00000000e+000  -1.00000000e+000  0.00000000e+000
vertex 3  xyz 1.00000000e+000  1.00000000e+000  0.00000000e+000
vertex 4  xyz -1.00000000e+000  1.00000000e+000  0.00000000e+000
vertex 5  xyz -5.00000000e-001  0.00000000e+000  0.00000000e+000
vertex 6  xyz 5.00000000e-001  0.00000000e+000  0.00000000e+000
vertex 7  xyz 0.00000000e+000  0.00000000e+000  0.00000000e+000
vertex 8  xyz 0.00000000e+000  0.00000000e+000  0.00000000e+000  coincide vertex 7

curve 1  order 2  vertex 1  2
curve 2  order 2  vertex 2  3
```

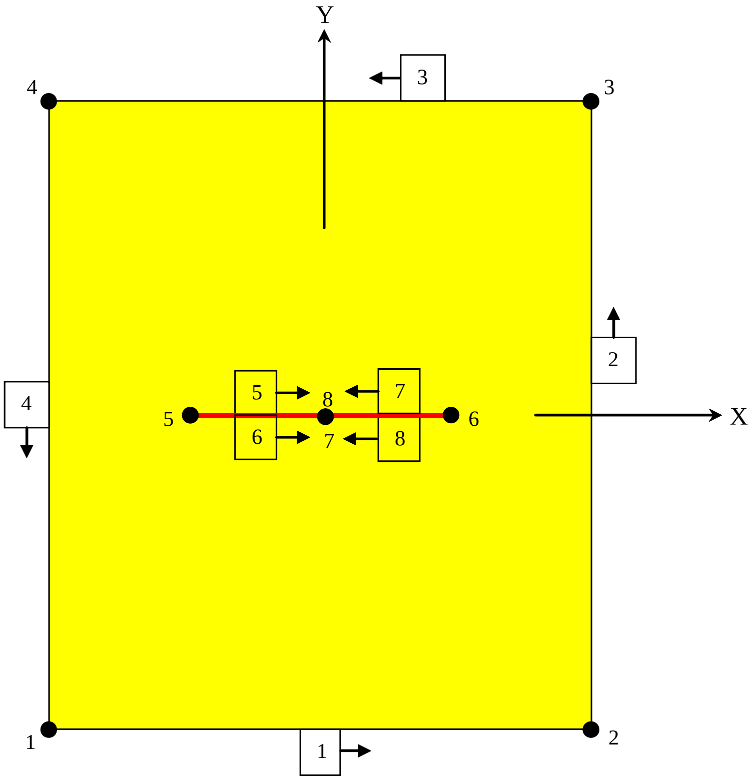


Figure C.5: Matrix, Crack inclusion with interface elements

```
curve 3  order 2  vertex 3  4
curve 4  order 2  vertex 4  1
curve 5  order 2  vertex 5  8
curve 6  order 2  vertex 5  7  coincide curve 5
curve 7  order 2  vertex 6  8
curve 8  order 2  vertex 6  7  coincide curve 7

patch 1 normal 0.00000000e+000  0.00000000e+000  1.00000000e+000  boundary curve 1 2 3 4 \
     boundary curve -5 6 -8 7   size 1.000e-001  property 1

crack NLFM
crack 1 structure curve 5 6 front2d 5
crack 2 structure curve 8 7 front2d 6
crack structure path 1 property 2
crack structure path 2 property 2

vertex 5000 xyz -5.00000000e-001  0.00000000e+000  0.00000000e+000  coincide vertex 5 slave node
vertex 5001 xyz 0.00000000e+000  0.00000000e+000  0.00000000e+000  coincide vertex 8 slave node
vertex 5002 xyz -5.00000000e-001  0.00000000e+000  0.00000000e+000  coincide vertex 5 slave node
vertex 5003 xyz 0.00000000e+000  0.00000000e+000  0.00000000e+000  coincide vertex 7 slave node
vertex 5004 xyz 5.00000000e-001  0.00000000e+000  0.00000000e+000  coincide vertex 6 slave node
vertex 5005 xyz 0.00000000e+000  0.00000000e+000  0.00000000e+000  coincide vertex 7 slave node
vertex 5006 xyz 5.00000000e-001  0.00000000e+000  0.00000000e+000  coincide vertex 6 slave node
vertex 5007 xyz 0.00000000e+000  0.00000000e+000  0.00000000e+000  coincide vertex 8 slave node

curve 5000 vertex 5000  5001  order 2  coincide curve 5 slave element
curve 5001 vertex 5002  5003  order 2  coincide curve 6 slave element
curve 5002 vertex 5004  5005  order 2  coincide curve 8 slave element
curve 5003 vertex 5006  5007  order 2  coincide curve 7 slave element
```

Notice that the single interior crack is actually composed of two cracks. Curves 5/6 form one side of the crack, while curves 7/8 form the other side. Vertices 5 and 6 serve as the crack fronts, and vertices 7 and 8 (coinciding vertices) form the middle of the crack. An interior crack must have these coinciding vertices at its center. This boundary description contains only one patch, but this patch contains two definitions of boundary curves. The first set of boundary curves defines the edges of the matrix, while the second set of boundary curves defines the crack.

C.6 Sphere

```
##########
# sphere #
##########
# Comments from Daniel
# I would recommend you to use two surfaces, each for one hemisphere,
# and each one obtained by rotating cubic circular 180 degree arc
# by angle of 180 degree using cubic expansion in the direction of revolution.
# Here is the example of a sphere centered at origin with radius equal to one:

# t3d -d 0.1

vertex 1 xyz 1 0 0
vertex 2 xyz -1 0 0

curve 1 vertex 1 2 order 4
```

```
polygon 1 xyz 1 2 0 weight 0.3333333333333
polygon 2 xyz -1 2 0 weight 0.3333333333333

curve 2 vertex 1 2 order 4
polygon 1 xyz 1 -2 0 weight 0.3333333333333
polygon 2 xyz -1 -2 0 weight 0.3333333333333

# collapsed curve defined as ordinary 180 deg circular arc
curve 3 vertex 1 1 order 4
polygon 1 poly 0 weight 0.3333333333333
polygon 2 poly 0 weight 0.3333333333333

# collapsed curve defined as ordinary 180 deg circular arc
curve 4 vertex 2 2 order 4
polygon 1 poly 0 weight 0.3333333333333
polygon 2 poly 0 weight 0.3333333333333

surface 1 curve 1 3 2 4
# polygon of a 180 deg circular arc from polygon point 1 of curve 1
# to polygon point 1 of curve 2 in plane yz+
# weights of this new polygon are obtained by multiplication of weights
# corresponding to 180 deg circular arc by the weight of appropriate
# polygon point on revoluted curve (1/3 * 1/3)
polygon 1 1 xyz 1 2 4 weight 0.111111111111
polygon 2 1 xyz -1 2 4 weight 0.111111111111
# polygon of a 180 deg circular arc from polygon point 2 of curve 1
# to polygon point 2 of curve 2 in plane yz+
polygon 1 2 xyz 1 -2 4 weight 0.111111111111
polygon 2 2 xyz -1 -2 4 weight 0.111111111111
# ordering of curves is different from surface 1 in order to get the
# normal pointing out of the sphere (but it is not necessary to ensure that)

surface 2 curve 2 4 1 3
# polygon of a 180 deg circular arc from polygon point 2 of curve 2
# to polygon point 2 of curve 1 in plane yz-
polygon 1 1 xyz -1 -2 -4 weight 0.111111111111
polygon 2 1 xyz 1 -2 -4 weight 0.111111111111
# polygon of a 180 deg circular arc from polygon point 1 of curve 2
# to polygon point 1 of curve 1 in plane yz-
polygon 1 2 xyz -1 2 -4 weight 0.111111111111
polygon 2 2 xyz 1 2 -4 weight 0.111111111111
```