CalculiX USER'S MANUAL
- CalculiX GraphiX, Version 2.23 -

Klaus Wittig

February 19, 2026

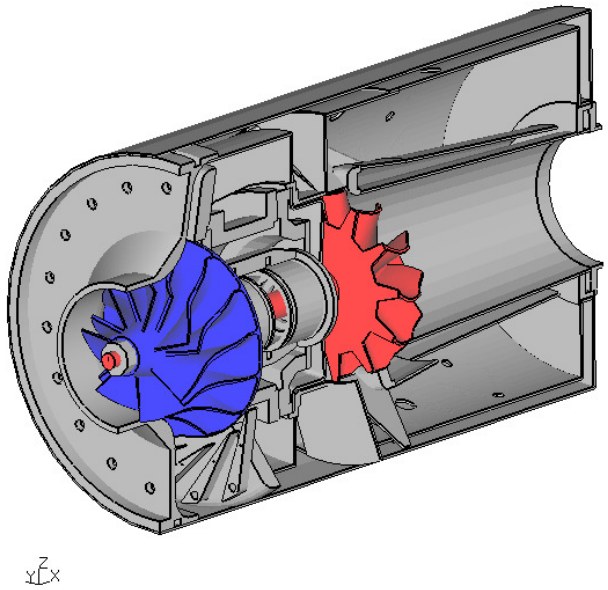


Figure 1: A complex model made from scratch using second order brick elements

Contents

1 Introduction **8**

2 Concept **8**

3 File Formats **10**

4 Getting Started **11**

5 Program Parameters **15**

6 Input Devices **17**
6.1 Mouse . . . 17
6.2 Keyboard . . . 18

7 Menu **18**
7.1 Datasets . . . 18
7.1.1 Entity . . . 19
7.2 Viewing . . . 19
7.2.1 Show Elements With Light . . . 20
7.2.2 Show Bad Elements . . . 20
7.2.3 Fill . . . 20
7.2.4 Lines . . . 20
7.2.5 Dots . . . 20
7.2.6 Flip shell elements . . . 20
7.2.7 Toggle Culling Back/Front . . . 20
7.2.8 Toggle Illuminate Backface . . . 20
7.2.9 Toggle Model Edges . . . 21
7.2.10 Toggle Element Edges . . . 21
7.2.11 Toggle Surfaces/Volumes . . . 21
7.2.12 Toggle Move-Z/Zoom . . . 21
7.2.13 Toggle Background Color . . . 21
7.2.14 Toggle Vector-Plot . . . 21
7.2.15 Toggle Add-Displacement . . . 22
7.2.16 Toggle Shaded Result . . . 22
7.2.17 Toggle Transparency . . . 22
7.2.18 Toggle Ruler . . . 22
7.2.19 Colormap . . . 22
7.3 Animate . . . 22
7.3.1 Start . . . 22
7.3.2 Tune-Value . . . 23
7.3.3 Steps per Period . . . 23
7.3.4 Time per Period . . . 23
7.3.5 Toggle Real Displacements . . . 23
7.3.6 Toggle Static Model Edges . . . 23
7.3.7 Toggle Static Element Edges . . . 23

7.3.8 Toggle Dataset Sequence . . . 23
7.4 Frame . . . 23
7.5 Zoom . . . 24
7.6 Center . . . 24
7.7 Enquire . . . 24
7.8 Cut . . . 24
7.9 Graph . . . 24
7.10 User . . . 25
7.11 Orientation . . . 25
7.11.1 +x View . . . 25
7.11.2 -x View . . . 25
7.11.3 +y View . . . 25
7.11.4 -y View . . . 25
7.11.5 +z View . . . 25
7.11.6 -z View . . . 25
7.12 Hardcopy . . . 25
7.12.1 Tga-Hardcopy . . . 25
7.12.2 Ps-Hardcopy . . . 25
7.12.3 Gif-Hardcopy . . . 26
7.12.4 Png-Hardcopy . . . 26
7.12.5 Start Recording Gif-Movie . . . 26
7.13 Help . . . 26
7.14 Toggle CommandLine . . . 26
7.15 Quit . . . 26

8 Customization **27**

9 Commands **27**
9.1 anim . . . 28
9.2 area . . . 28
9.3 asgn . . . 29
9.4 aver . . . 32
9.5 bia . . . 32
9.6 body . . . 33
9.7 break . . . 34
9.8 call . . . 34
9.9 capt . . . 34
9.10 cmap . . . 34
9.11 cntr . . . 35
9.12 col . . . 35
9.13 comp . . . 35
9.14 cont . . . 36
9.15 copy . . . 36
9.16 corrad . . . 38
9.17 csysa . . . 38
9.18 cut . . . 39

9.19 del . . . 39
9.20 dist . . . 40
9.21 div . . . 41
9.22 ds . . . 42
9.23 elem . . . 45
9.24 else . . . 46
9.25 else if . . . 46
9.26 elty . . . 46
9.27 endif . . . 48
9.28 endwhile . . . 48
9.29 enq . . . 48
9.30 eprop . . . 50
9.31 eqal . . . 50
9.32 exit . . . 51
9.33 fil . . . 51
9.34 flip . . . 51
9.35 flpc . . . 51
9.36 font . . . 51
9.37 frame . . . 52
9.38 gbod . . . 52
9.39 gonly . . . 52
9.40 graph . . . 52
9.41 grpa . . . 54
9.42 grps . . . 55
9.43 gsur . . . 55
9.44 gtol . . . 55
9.45 hcpy . . . 56
9.46 help . . . 56
9.47 if . . . 57
9.48 int . . . 57
9.49 init . . . 57
9.50 lcmb . . . 58
9.51 length . . . 58
9.52 line . . . 58
9.53 lnor . . . 59
9.54 mata . . . 59
9.55 map . . . 60
9.56 mats . . . 61
9.57 max . . . 61
9.58 maxc . . . 61
9.59 maxr . . . 61
9.60 menu . . . 61
9.61 merg . . . 62
9.62 mesh . . . 62
9.63 mids . . . 63
9.64 min . . . 63

9.65 minc . . . 64
9.66 minr . . . 64
9.67 minus . . . 64
9.68 mm . . . 64
9.69 move . . . 65
9.70 movi . . . 66
9.71 msg . . . 67
9.72 mshp . . . 67
9.73 neigh . . . 68
9.74 node . . . 71
9.75 norm . . . 71
9.76 nurl . . . 71
9.77 nurs . . . 72
9.78 ori . . . 73
9.79 plot . . . 73
9.80 plus . . . 75
9.81 pnt . . . 76
9.82 prnt . . . 76
9.83 proj . . . 79
9.84 qadd . . . 80
9.85 qali . . . 81
9.86 qbia . . . 81
9.87 qbod . . . 81
9.88 qcnt . . . 82
9.89 qcut . . . 82
9.90 qdel . . . 83
9.91 qdis . . . 84
9.92 qdiv . . . 84
9.93 qenq . . . 85
9.94 qfil . . . 85
9.95 qflp . . . 86
9.96 qint . . . 87
9.97 qlin . . . 88
9.98 qmov . . . 89
9.99 qmsh . . . 89
9.100qnor . . . 90
9.101qpnt . . . 91
9.102qnod . . . 91
9.103qrem . . . 91
9.104qseq . . . 92
9.105qshp . . . 92
9.106qspl . . . 93
9.107qsur . . . 93
9.108qtxt . . . 94
9.109quit . . . 95
9.110read . . . 95

9.111rep . . . 99
9.112rnam . . . 99
9.113rot . . . 99
9.114save . . . 100
9.115scal . . . 100
9.116send . . . 101
9.117seqa . . . 116
9.118seqc . . . 116
9.119seql . . . 116
9.120seta . . . 116
9.121setc . . . 118
9.122sete . . . 119
9.123seti . . . 119
9.124setl . . . 119
9.125seto . . . 119
9.126setr . . . 120
9.127shpe . . . 120
9.128split . . . 121
9.129stack . . . 123
9.130steps . . . 123
9.131stop . . . 123
9.132subm . . . 124
9.133surf . . . 124
9.134swep . . . 124
9.135sys . . . 126
9.136test . . . 127
9.137thrs . . . 127
9.138tra . . . 128
9.139trfm . . . 128
9.140txt . . . 129
9.141typs . . . 131
9.142ucut . . . 131
9.143ulin . . . 131
9.144val . . . 131
9.145valu . . . 132
9.146view . . . 134
9.147volu . . . 135
9.148while . . . 135
9.149wpos . . . 136
9.150wsize . . . 136
9.151zap . . . 136
9.152zoom . . . 136

10 Element Types **138**

11 Result Format **144**
11.1 Model Header Record . . . 145
11.2 User Header Record . . . 145
11.3 Nodal Point Coordinate Block . . . 145
11.4 Element Definition Block . . . 146
11.5 Parameter Header Record . . . 147
11.6 Nodal Results Block . . . 148

12 Pre-defined Calculations **150**
12.1 Von Mises Equivalent Stress . . . 151
12.2 Signed Von Mises Equivalent Stress . . . 151
12.3 Von Mises Equivalent Strain . . . 151
12.4 Signed Von Mises Equivalent Strain . . . 151
12.5 Principal Stresses . . . 151
12.6 Principal Strains . . . 151
12.7 maxShear Stresses . . . 152
12.8 Contact Stresses . . . 152
12.9 Contact Strain . . . 152
12.10 Cylindrical Stresses . . . 152
12.11 Weighted Error . . . 152

13 Meshing rules **152**

14 User-Functions **154**

15 Internal Setnames **154**

A Known Problems **156**
A.1 Program does not show the geometry after startup . . . 156
A.2 Program is not responding . . . 156
A.3 Program generates a segmentation fault . . . 156

B Tips and Hints **156**
B.1 How to change the format of the movie file . . . 156
B.2 How to get the sets from a geo- or ccx-inp file for post-processing 157
B.3 How to define a set of entities . . . 157
B.4 How to enquire node numbers and values at certain locations . . 158
B.5 How to select only nodes on the surface . . . 158
B.6 How to write values to a file . . . 158
B.7 How to generate a user dataset . . . 159
B.8 How to generate a time-history plot . . . 159
B.9 How the mesh is related to the geometry . . . 160
B.10 How to change the order of elements . . . 161
B.11 How to connect independent meshes . . . 161
B.12 How to define loads and constraints . . . 161
B.13 How to map loads . . . 162

B.14 How to run cgx in batch mode 164
B.15 How to process results 165
B.16 How to deal with CAD-geometry 165
B.17 How to check an input file for ccx 171
B.18 Remarks Concerning Ansys 173
B.19 Remarks Concerning Code Aster 173
B.20 Remarks Concerning dolfyn 174
B.21 Remarks Concerning Duns and Isaac 174
B.22 Remarks Concerning ls-dyna 175
B.23 Remarks Concerning Nastran 176
B.24 Remarks Concerning NETGEN 176
B.25 Remarks Concerning OpenFOAM 177
B.26 Remarks Concerning Samcef 177

C Simple Examples **179**
C.1 Disc 179
C.2 Cylinder 181
C.3 Sphere 182
C.4 Sphere (Volume) 183
C.5 Sphere (Thin Volume) 184
C.6 Airfoil for cfd codes 187
C.7 If and while demo 192
C.8 Data storage in a user dataset 194
C.9 User File Parser 195

1 Introduction

This document is the description of CalculiX GraphiX (cgx). This program is designed to generate and display finite elements (FE) and results coming from CalculiX CrunchiX (ccx). If you have any problems using cgx, this document should solve them. If not, you might send an email to the author [3]. The Concept and File Format sections give some background on functionality and mesher capabilities. The Getting Started section describes how to run the verification examples you should have obtained along with the code of the program. You might use this section to check whether you installed CalculiX correctly. Then, a detailed overview is given of the menu and all the available keywords in alphabetical order in the Menu and Commands sections respectively. Finally, the User's Manual ends with the appendix and some references used while writing the code.

2 Concept

This program uses the openGL library for visualization and the glut library [2] for window management and event handling. This results in very high speed

if a hardware-accelerated openGL-library is available and still high speed for software-rendering (MesaGL,[1]).

The cgx has pre- and post-processor capabilities. It is able to generate and display beam, shell and brick elements in its linear and quadratic form (fig. 1), tets can be generated from within cgx if either TETGEN [5] or the program ng_vol (part of NETGEN [4]) is accessible (see also "How to deal with CAD-geometry"). TETGEN comes with cgx.

The built-in mesher creates a structured mesh based on a description of the geometry. For example, it uses lines for beam elements, surfaces for shell elements and volumes (bodies) for brick elements. The program distinguishes between the mesh and the underlying geometry. Elements are made from faces and faces are made from nodes. If you move a node, the corresponding face(s) and element(s) will follow. The geometry behaves according to the mesh: Lines are made from points, surfaces are made from lines and bodies are made of surfaces. As a result, if you modify the position of a point, all related geometry will follow. In other words, if the location of geometric entities is changed, it is necessary to move the points on which the entities rely. It should be noted that faces exist only on free surfaces of the model.

In addition, entities can be grouped together to make sets. Sets are useful to handle parts of a model. For example, sets can be used to manipulate or display a few entities at a time (see also "How to define a set of entities").

A simple but powerful entity which can store values (character strings) is also available. This values can be derived from previous commands or calculated results by using an internal stack. Simple calculations can be performed. The values can be used to substitute parameters of subsequent commands. The user might measure a distance or calculate a distance and use this value to move a part of the mesh. Together with a 'while' loop, an 'if' case distinguishing command and the possibility to use system calls via the 'sys' command, elaborated batch files can be written.

After a mesh is created in cgx, it needs written to a file for use with the solver. Likewise, several boundary conditions and loads can be written to files (see "How to connect independent meshes", "How to define loads and constraints" and "send"). These files need to be added into the control file for later use in ccx. Additional commands, material description and so on must be added with the help of an external editor. Existing fields like a pressure distribution or temperatures may be mapped from one mesh to another. Please see "How to map loads" how to do that.

After the analysis is completed, the results can be visualized by calling the cgx program again in an independent session. The program is primary controlled by the keyboard with individual commands for each function. Only a subset of commands which are most important for post-processing is also available through a pop-up menu. Shaded animations of static and dynamic results, the common color plots and time history plots can be created. The outer faces can be switched off to make inner structures like a cavity visible or certain faces may be displayed in a transparent manner. Also, a cut through the model can be done which creates a section and it is possible to zoom through the model.

Skilled users might include their own functions. For example someone may need his own functions to manipulate the result-data or he may need an interface to read or write his own results format (see "call").

Both the pre- and post- processing can be automated in batch-mode (see "How to run cgx in batch mode").

The program searches the home directoy for a file named ".cgx" (see "Customization"). The commands written there will be executed during startup. The user might store there "menu" commands which link user written command files to the menu or other personal settings like "view cl" to switch the command line from the konsole to the graphic's window. Also the helpfiles, the browser and the psviewer (for graphs) can be redefined (see "asgn").

3 File Formats

It is hoped by the author that common CAD formats will be supported by stand-alone interfaces which translate into fbd-commands. So far vda, step and iges to fbd interfaces are available on the CalculiX home pages. Tet-meshes can be generated based on the resulting fbd-files. The following file-formats are available to write(w) and/or read(r) geometric entities:

- fbd-format(r/w), this format consists of a collection of commands explained in the section "Commands" and it is used to store geometrical information like points, lines, surfaces and bodies. All geometry generated by the user is stored in this format. But it can also be used to define a batch job which uses the available commands.

- step-format(r), reverse engineered based on some cad files. Only points and certain types of lines are supported currently. Be aware of the more powefull cad2fbd interface program on the CalculiX home page.

- stl-format in ascii (r/w) and binary, this format describes a shape using only triangles.

The following file-formats are available to write a mesh and certain boundary-conditions:

- Abaqus, which is used by the CalculiX solver ccx.

- Ansys, several boundary conditions available.

- Code_Aster, mesh and sets of nodes and elements are available.

- dolfyn, a free cfd-code [6].

- duns, a free cfd-code [7].

- isaac, a free cfd-code [8].

- ls-dyna, mesh, node- and element sets and surfaces.

- Nastran, several boundary conditions available.
- OpenFOAM, a free cfd-code [9], only 8-noded brick-elements are supported.
- Samcef, mesh and sets of nodes and elements are available.
- tochnog, a free fem-code [10], only 8-noded brick-elements are supported.

The following solver-input-file-formats can be read to check the mesh, sets and certain boundary-conditions:

- Abaqus, this is also used by the CalculiX solver ccx.
- Netgen, read Netgen native format (.vol)

The following file-formats are available to read solver results:

- frd-format, files of this format are used to read results of previous calculations like displacements and stresses. This format is described in section "Result Format." It is used by the CalculiX solver ccx.
- duns, a free cfd-code [7],
- isaac, a free cfd-code [8],
- OpenFOAM, a free cfd-code [9].
- Nastran, the f06-file can be read (sf. only CHEXA, displacements and stresses). Unfortunatelly this format differs from version to version and has to be adapted occasionally.
- vtk, so far only the mesh without results (nodes, qu4 and tet4 elements).

For a more detailed description on how to use cgx to read this formats see the "read" command, "Program Parameters" and the program specific "Tips and Hints" sections. See the "send" command for how to write them from cgx.

4 Getting Started

For installation help, see .../Calculix/cgx_X.X/INSTALL. Be aware of the possibility to redefine the helpfiles, browser and psviewer which is explained there.

After the program is installed on your machine, you should check the functionality by running the examples included in the distribution. The examples are located in .../Calculix/cgx_X.X/examples/. Begin with a result file called result.frd. Just type

"cgx result.frd"

and some information is echoed in the konsole and a new window called main

window appears on the screen. The name conventions used for the different areas in the main-window are explained in figure 2. Now you should move the mouse pointer into the menu-area and press the left mouse-button. Keep it pressed and continue over the menu item “Dataset” to “Disp”. There you release the button. Then press the left button again and continue over “Dataset” and “Entity” to “D1”. For background informations look into the subsection ”Datasets” and ”Entity” which explains how to display results. After seeing the values you might play around a bit with the ”Menu”. Before going further, you should read the section ”Input Devices”. See also the commands ”steps”, ”maxr”, ”minr”, ”max”, ”min” (or the combination of max and min ”mm”) and ”scal” which might be used to modify the colour representation of the displayed values. For example type “min 0” to set the lower value of the colour bar to zero. Now you should study the following interactive commands: Use ”qenq” to enquire values at nodes. Use ”qtxt” to generate node attached texts showing their number and value. Use ”capt” to change the displayed filename to a user defined text and ”ulin” to add a second line under the filename. Use ”qcut” to generate a section through the model. And use ”graph” to generate a 2D time history plot (for results with several time-steps) or a 2D plot of values along a sequence of nodes (see ”qseq”). In case you want to display just a set of nodes, faces or elements with their results use the ”plot”, ”plus” and ”minus”. commands (see also ”How to get the sets from a geo- or ccx-inp file for post-processing”). Watch out when you type a command; the cgx window MUST stay active and not the konsole from which the program was started. It is better to stay with the mouse pointer in the cgx window. Next, ”Quit” the program and type

”cgx -b geometry.fbd”

in the konsole. The program starts again but now you see only a wire-frame of the geometry. Move the mouse-pointer into the new window and type ”mesh all”. The mouse-pointer MUST stay in this window during typing and NOT in the konsole from which the program was started. After you see ”ready” in the parent konsole, the mesh is created. To actually see it, type ”plus ea all”. Now you see the mesh in green color. To see the mesh as a wire-frame, choose in the main menu”Viewing” and continue to the entry ”Toggle Element Edges” and then again in ”Viewing” choose ”Dots”. To see the mesh illuminated chose in the main menu ”Viewing” and continue to the entry ”Show Elements With Light”. To see it filled, choose in the main menu ”Viewing” and continue to the entity ”Fill”. Most of the time it is sufficient to see the surface elements only. For this purpose, choose in the main menu ”Viewing” and continue to the entry ”Toggle Surfaces/Volumes”. If you start cgx in the post processor mode, as you did in the first example (cgx result.frd), the surface mode is automatically set. To see the interior of the structure, choose in the main menu ”Viewing” and continue to the entity ”Toggle Culling Back/Front”. To save the mesh in the format used by the solver, type ”send all abq”. To store the mesh in the result format type ”send all frd”.

To create a new model start the cgx by typing

"cgx -b file"

where "file" will be the name of the new model if you later exit the program with the command "exit". The way to create a model from scratch is roughly as follows, create

- points with "qpnt" or "pnt",
- lines with "qlin",
- surfaces with"qsur",
- Bodies with "qbod".

If possible, create higher geometry by sweeping or copying geometry with "swep" or "copy". You might move or scale your model with the command "move". The commands require sets to work with. Sets reference entities like bodies or nodes. They are useful because you can deal with a bunch of entities at once. See the section "How to define a set of entities" about how to create them. You may write a file with basic commands like "pnt" to create the basis for your construction and read it with the "read" command. Most commands can be used in batch mode. This allows the user to write a command file for repeated actions.

If you want to start with CAD geometry then please read "How to deal with CAD-geometry".

The interactive commands start with the letter 'q'. Please make yourself familiar with all of them before you start to model complex geometry.

After the geometry is created, the divisions of the lines can be changed to control the density of the elements. Display the lines and their divisions with

- "plot ld all".

To change the element division, use

- "qdiv".

The default division is "4". With a division of"4," a line will have 6 nodes and will therefore be the edge of two element of the quadratic type. Next, the type of the elements must be defined. This can be done for each of the different sets. A new assignment will replace a previous one. Delete all previous assignments with

- "elty all"

and assign new types with

- "elty all he20".

If a mesh is already defined type

- "del mesh"

and mesh again with

- "mesh all".

Then choose the menu entity "Viewing - Show Elements With Light" to see the illuminated mesh. Lastly, export the mesh in the calculix solver format with

- "send all abq".

With the "send" command, it is also possible to write boundary conditions, loads and equations to files. The equations are useful to "glue" parts together (see "How to connect independent meshes".

It is advisable to save your work from time to time without exiting the program. This is done with the command

- "save".

You leave the program either with

- "exit"

or with

- "quit".

Exit will write all geometry to an fbd-file and if a file of this name exists already then the extension of this file will be renamed from fbd to fbb. "quit" closes the program without saving.

A solver input file can be written with the help of an editor (emacs, nedit etc.). If you write a ccx command file, then include the mesh, the boundary conditions etc. with the ccx command "*INCLUDE". After you finished your input-file for the solver (ccx) you might read it by calling the program again with

"cgx -c solverfile.inp"

for a final check. All predefined sets are available together with automatically generated sets which store boundaries, equations and more. These sets start with the "+"-sign. For example the set +bou stores all constrained nodes where the set +bou1, +bou2, +bou3 store the constraints for the individual directions. Further the set +dep and +ind store the dependent and independent nodes involved in equations etc. See which sets are defined with the command

- "prnt se".

Each line starts with the set-index, then the set-name followed by the number of all referenced entities. The sets can be specified by index or name. For example if the index of set "blade" is "5" the following commands are equivalent:

- "plot p 5"

- "plot p blade"

The use of wildcards is possible to search for a certain expression:

- "prnt se +*"

Now all sets starting with a "+" in their names will be listed.

Predefined loads are stored as "Datasets" to be visualized. Sets with the name of the load-type (CLOAD, DLOAD) store the related nodes, faces or elements. Use the command

- "plot"

or

- "plus"

to visualize entities of sets.

Then run the input file with ccx. The result file (.frd) can be visualized with

"cgx result.frd solverfile.inp"

were the solver input file "filename.inp" is optional. With this file, the sets, boundary conditions and loads used in the calculation are available together with the results.

If you have problems doing the above or if you want to learn more and in more detail about the cgx continue with the tutorial [11] and look in the appendix, section Tips and Hints and Known Problems.

Eventually adapt the ".cgx" file in your home directory to your preferences. Especially configure the help system and the psviewer for the "graph" command if necessary (see "Customization").

5 Program Parameters

```
usage:
cgx [-a|-b|-bg|-c|-duns2d|-duns3d|-isaac2d|-isaac3d|-foam|-ng|
     -step|-stl] filename [ccxfile]

 -a        automatic-build-mode, geometry file derived from a
           cad file is expected
 -b        build-mode, geometry file in fbd-format is expected
 -bg       background, suppress creation of graphic output
           otherwhise as -b, geometry (command) file must be
           provided
 -c        read an solver input file (ccx, Abaqus)
 -duns2d   read duns result files (2D)
 -duns3d   read duns result files (3D)
```

```
 -duns2dl   read duns result files (2D, long format)
 -duns3dl   read duns result files (3D, long format)
 -isaac2d  read isaac result files (2D)
 -isaac3d  read isaac result files (3D)
 -foam     read the OpenFOAM result directory structure
 -f06      read Nastran f06 file.
 -ng       read Netgen native format (with surface domains)
 -step     read an ascii-step file (points and lines only)
 -stepsplit read step and write its parts to the filesystem
            in separate directories
 -stl      read an stl file (triangles)
 -vtk      read an vtk file (tet4, qu4 w/o results)
 [-v]      (default) read a result file in frd-format and
           optional a solver input file (ccx) in addition
           which provides the sets and loads used in the
           calculation.

special purpose options:
 -mksets       make node-sets from *DLOAD-values
               (setname:''_<value>'')
 -read         forces the program to read the complete result-
               file at startup
```

If no option is provided then a result-file (frd) is assumed, see "Result Format".

A file containing commands or geometric informations is assumed if the option -b is specified. Such a file will be created if you use "exit" or "save" after you have interactively created geometry. Option -a awaits the same format as option -b but merging, defining of line-divisions and the calculation of the interior of the surfaces is done automatically and the illuminated structure is presented after startup. This should be used if the command file was generated by an interface program which convertes cad-data to cgx-format (for example vda2fbd). With option -a and -b the program will start also if no file is specified.

An input file for the solver can be read with option -c. Certain key-words are known and the affected nodes or elements are stored in sets. For example the default set(s) +bou(dof) store nodes which are restricted in the corresponding degree of freedom and the set(s) +dep(dof) and +ind(dof) store dependent and independent nodes used in equations.

A special case is OpenFOAM. The results are organized in a directory structure consisting of a case containing time-directories in which the result-files are stored. The user must call cgx using the case-directory (cgx -foam case). The program will then search the time-directories. The time directories must contain a time-file to be recognized. Or in other words each directory in this level containing a time-file is regarded as a result directory.

6 Input Devices

6.1 Mouse

The mouse is used to manipulate the view-point and scaling of the object inside the drawing area (figure 2). Rotation of the object is controlled by the left mouse button, zoom in and out by the middle mouse button and translation of the object is controlled by the right mouse button. Inside the menu area, the mouse triggers the main menu with the left button.

In addition the mouse controls the animation of nodal values. The animation will stop if the mouse pointer is not in the drawing area but will start again if the pointer enters the drawing area. This can be prevented by pressing the middle mouse button while the mouse pointer is in the menu area. Pressing the right button will release the next frame. A frozen animation can be released by pressing the middle button. The previous frame can be reloaded by pressing the middle mouse button twice and the right button once (while the mouse is in the menu area).

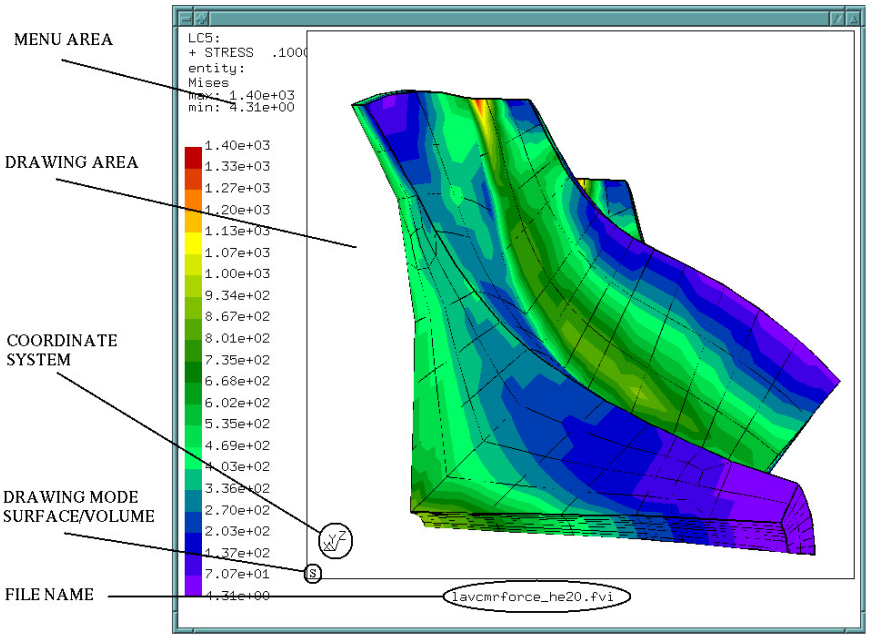


Figure 2: structure of the main-window

6.2 Keyboard

The Keyboard is used for command line input and specifying the type of entities when selecting them with the mouse pointer. The command line is preferable in situations where pure mouse operation is not convenient (i.e. to define a certain value) or for batch controlled operations. Therefore most commands are only available over the command line. The stream coming from the keyboard is echoed in the parent-konsole but during typing the mouse pointer must stay inside the main window. Otherwise the commands will not be recognized by the program. The user might use the menu fuction “Toggle CommandLine” or the command “view cl” to switch the command line from the konsole to the graphic's window.

The following special keys are used:

```
Special Keys:
 ARROW_UP:   previous command
 ARROW_DOWN: next command
 PAGE_UP:    entities of previous set (if the last command was
             plot or plus) or the previous Loadcase
 PAGE_DOWN:  entities of next set (if the last command was
             plot or plus) or the next Loadcase
```

7 Menu

The main menu pops up when pressing the left mouse-button inside the menu-area (figure 3). It should be noted that there are equivalent command-line functions for most of the menu-functions. This can be used for batch-controlled post-processing (see command "view". Next the entities inside the main menu will be explained:

7.1 Datasets

Datasets are selected with the menu-item "Dataset". A dataset is a block of nodal values. These could be displacements due to a linear analysis or for a specific time-step during a nonlinear analysis. It could also contain other values like stresses, strains, temperatures or something else. To select a dataset, make sure that the mouse-pointer is inside the menu area. Then, press the left mouse button and move the mouse-pointer over the menu entry "Dataset", then continue to the right. A sub-menu pops up showing all available datasets. Each dataset entry comprises the step number (NUMSTP: see "Nodal Results Block") followed by the name (NAME), a dataset-value (VALUE, usually time or frequency) and a dataset-description (TEXT). Move the mouse-pointer over a dataset you are interested in and release the left mouse button. The dataset is now selected. A results "Entity" must be chosen to see the values in the drawing-area. This Dataset might also contain automatically calculated values like the v. Mises stress and the maximum principal stress (see Pre-defined

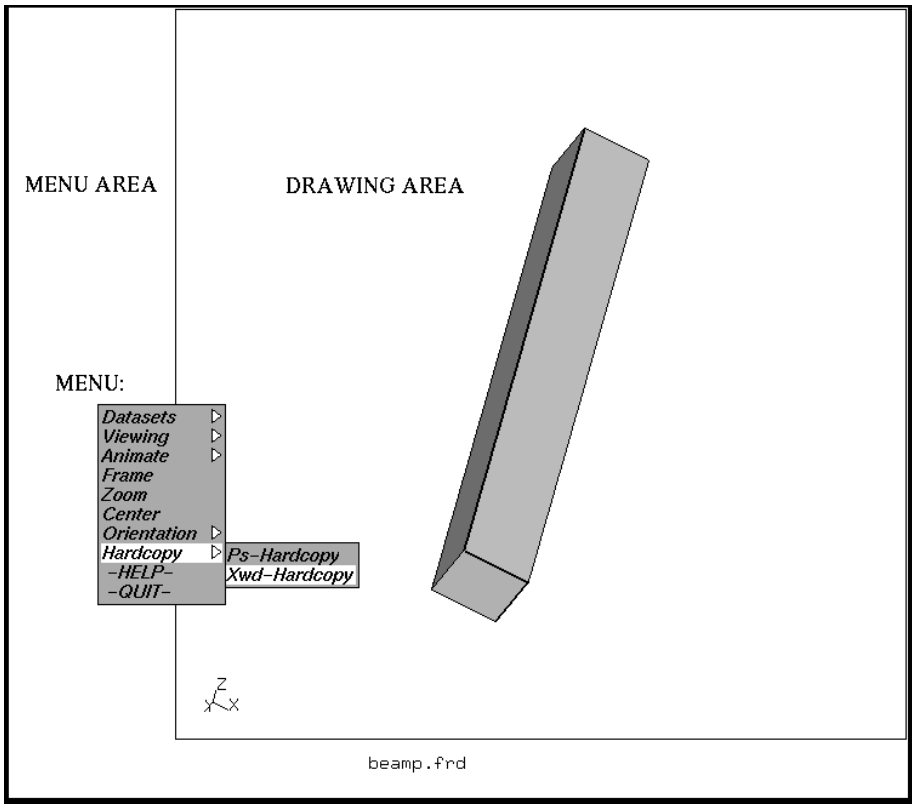


Figure 3: structure of the main-menu

Calculations and Result Format). See also the command "ds" to control the functionality with the command-line.

7.1.1 Entity

To view data from the dataset, its also necessary to specify the entity (i.e. dx for a displacement Dataset). It works in the same way as for selecting the dataset but instead of releasing the left mouse button over a Dataset continue to the sub-menu "Entity." Continue from that item to the right and release the mouse button when the pointer is over the desired entity. Now the data will be displayed in the drawing-area.

7.2 Viewing

In the following sections, changing properties and styles of the displayed structure are explained. See the command "view" to control the functions with the command-line.

7.2.1 Show Elements With Light

This is the default view of the mesh if the program was started in viewing mode. If used, any animation will be interrupted and no values are displayed.

7.2.2 Show Bad Elements

This option presents elements which have a negative Jacobian value at least at one integration point. The solver ccx can not deal with those elements. So far, only TET and HEX elements are checked. These elements are stored in the set called -NJBY. See also the command "eqal".

7.2.3 Fill

This is the default mode and forces the element faces to be rendered.

7.2.4 Lines

The edges of the element faces are displayed. This is especially useful to see into the structure to find hot spots in the displayed field. With "Toggle Move-Z/Zoom" and "qcut", a more detailed analysis can follow. For very dense meshes switch to "Dots".

7.2.5 Dots

The corners of the element faces will be displayed. This is especially useful if values inside the structure need checked.

7.2.6 Flip shell elements

This function inverts the front and back side of shell elements. Since it is slow the user may work with "Toggle Culling Back/Front" which has often the desired effect without waiting time.

7.2.7 Toggle Culling Back/Front

This removes the faces of volume elements for all elements or for the surface of the structure, depending on the state of "Toggle Surfaces/Volumes". With this option, the user can visualize internal structures like cracks or a core of a hollow structure.

7.2.8 Toggle Illuminate Backface

Initially only the front faces are illuminated and the back faces are dark. This is helpful to determine the orientation of the elements. If you want to see all faces illuminated regardless of the orientation, then use this option. If you want to change the orientation of elements use the command "qflp".

7.2.9 Toggle Model Edges

Per default, all free element edges are shown. The user can remove/show them with this option.

7.2.10 Toggle Element Edges

Per default, just the free element edges are shown. The user might add all edges to the structure with that option.

7.2.11 Toggle Surfaces/Volumes

This switches the way each volume elements are displayed. Either all faces of the elements or just the element faces on the surface of the structure are displayed. Depending on the state of "Toggle Culling Back/Front," either the faces pointing to the user or the faces pointing away are displayed. The default is just to show the surface pointing to the user. In the lower left corner of the drawing area,(see figure 2) a character is printed, indicating the program is in the surface mode "s" or in the volume mode "v".

7.2.12 Toggle Move-Z/Zoom

Instead of zooming in with the help of the middle mouse button, it is also possible to move a clipping plane through the structure to get a view of the inside. The clipping plane is parallel to the screen and will be moved in the direction to and from the user by pressing the middle mouse button and moving the pointer up and down while inside the drawing area. Initially the clipping plane is located in the middle of the structure. The number below the axis cross states the actual position of the cutting plane relative to the middle of the structure in model dimensions. Consider using the "plot" and "plus" commands to customize your view. Depending on hardware, this functionality could be slow.

7.2.13 Toggle Background Color

With this option, it is possible to switch between a black and a white background.

7.2.14 Toggle Vector-Plot

It is possible to add small "needles" to the plot which point with their heads in the direction of the vectors. Only entities which are marked in the database as vectors will be affected. See "Nodal Results Block" for information on how entities are marked as vectors. Internally calculated vector-results, like the worst principal stress, are marked automatically. If one component or the value of a vector is selected, then the option takes immediate effect.

This option can be used in combination with "Animate Toggle Dataset Sequence".

See also the keyboard command "ds" how to select datasets and entities with the keyboard. In this case, entities which are NOT marked in the dataset as vectors can be displayed with vector-needles. This command line approach with "ds" is the only way to display isaac or duns-cfd-results with vector-needles. See also the command "scal" how to manipulate the length of the vectors.

7.2.15 Toggle Add-Displacement

It is possible to display results on the deformed structure. For example, you can display a stress field on the deformed structure. If you know a suitable amplification factor for your displacements then use the "scal" command to issue this value but this can also be done later. Of course displacements for the Loadcase must be available.

7.2.16 Toggle Shaded Result

It is possible to display results with illumination. For example, you can display a stress field with a shaded appearance to have a better impression of the shape of the structure.

7.2.17 Toggle Transparency

Transparent display in the surface mode is switched on or off.

7.2.18 Toggle Ruler

Triggers the display of a ruler bar.

7.2.19 Colormap

Changes the colormap of the color scale. The default is 'classic'. See also "cmap".

7.3 Animate

This option allows the animation of displacements. See also "anim", "ds" and "scal" to use this functionality with the command-line.

It is possible to create this sequence from just one Dataset, see "Start". This is useful for displaying mode-shapes. See also "Toggle Dataset Sequence" to create a sequence from multiple Datasets to visualize dynamic responses.

7.3.1 Start

Creates a sequence of display-lists to visualize displacements (for example mode-shapes). The program recognizes displacements just by the name of the dataset. This name must start with the letters "DISP", otherwise the animation will not start (see "Nodal Results Block").

7.3.2 Tune-Value

Controls the amplitude of the animation. If "Toggle Real Displacements" was chosen before, the tune-value is equivalent to the amplification of the animation.

7.3.3 Steps per Period

Determines how many display lists for one period of animation will be used. If "Toggle Dataset Sequence" was chosen, then these number of display lists will be interpreted as one period (see Time per Period).

7.3.4 Time per Period

Determines how many seconds per period.

7.3.5 Toggle Real Displacements

To see the correct displacement of each node. The animation can be controlled with the help of the mouse.

7.3.6 Toggle Static Model Edges

The user can switch on additional undeformed model edges. This is useful for hardcopies were this edges give a reference to the undeformed shape.

7.3.7 Toggle Static Element Edges

The user can switch on additional undeformed element edges. This is useful for hardcopies were this edges give a reference to the undeformed shape.

7.3.8 Toggle Dataset Sequence

Creates a sequence of display-lists to visualize values of a sequence of Datasets. The Datasets must use the same type, for example only displacements or only stresses. To activate the animation, after you have selected "Toggle Dataset Sequence" choose the first Dataset to be displayed, then the second and then the last one. Finally choose the entity. The first two datasets define the spacing between the requested datasets and the third-one defines the last dataset to be displayed. The last two selections of datasets can be omitted. Then all datasets which use the same name, starting from the selected one, will be used. The command "ds" provides the same functionality.

7.4 Frame

Adjusts the drawing box.

7.5 Zoom

Use this command to zoom into a rectangular section of the window. After this option is chosen, use the mouse to select the opposite corners of a rectangle. The display will zoom in on the rectangular area. Note the rectangle is never shown on the screen (see also "zoom").

7.6 Center

Used to choose a new center point for the structure. After this option is chosen, pick either a node, a point or the corner of an entity. To easily find the element corners, the function "Toggle Element Edges" is triggered automatically (see also "qcnt").

7.7 Enquire

Used to investigate parameters like the value and the position of a certain node of the model. Pick a node after this option is chosen. To easily find the element corners, the function "Toggle Element Edges" is triggered automatically (see also "qenq").

7.8 Cut

Used to cut elements and to create a section of new elements and nodes. Either pick three nodes, or, in case a dataset-entity of a vector was already selected, use the menu entry "vector" and select just one node. The cutting plane is then determined by the direction of the selected vector (displacements, worstPS etc.). The three room directions x,y,z can also be used to define a plane normal to the selected direction. To suppress or enable the immediate ploting of the cut chose "switch plot" (switch ploting on/off) before you create the cut. This is necessary if more than one cut should be shown together. So first "switch plot" then create all the cuts except the last. Then "switch plot" again and create the last cut. Select "Uncut" to delete the sections and to display the model.

To easily find the element corners, the function "Toggle Element Edges" is triggered automatically (see also "qcut" and "cut")

7.9 Graph

Used to generate a 2D-plot. The option "Length" will provide a plot "value over distance between nodes". The option "Datasets" will provide a plot "value over Dataset-nr" and the option "Time" will provide "value over Time". For the later two options it is necessary to first create an animation with either the command "ds" or the menu option "Toggle Dataset Sequence" (see also "graph" and "How to generate a time-history plot"). To easily find the element corners, the function "Toggle Element Edges" is triggered automatically.

7.10 User

This menu item does not exist until the first "menu" command was executed. Each "menu" command adds a new user command to the menu. This "menu" commands are usually stored in a ".cgx" file in the home directory to link them to the menu during startup.

7.11 Orientation

7.11.1 +x View

To look along the x-axis.

7.11.2 -x View

To look against the x-axis.

7.11.3 +y View

To look along the y-axis.

7.11.4 -y View

To look against the y-axis.

7.11.5 +z View

To look along the z-axis.

7.11.6 -z View

To look against the z-axis.

7.12 Hardcopy

To create a hard-copy during animation, it is useful to stop the animation first with the middle mouse button while inside the menu area of the main window and then release one picture after the other with the right button until the desired amplitude or step is reached.

7.12.1 Tga-Hardcopy

To create a window dump in tga format. You might use the program "convert" [12] to convert this format to others.

7.12.2 Ps-Hardcopy

To create a window dump in postscript format. The program convert must be installed.

7.12.3 Gif-Hardcopy

To create a window dump in gif format. The program convert must be installed.

7.12.4 Png-Hardcopy

To create a window dump in png format. The program convert must be installed.

7.12.5 Start Recording Gif-Movie

All frames during an animation are stored. The recording ends after the right mouse button is pressed while in the menu area. Finally all frames are combined in the file "movie.gif" which can be displayed with various tools (Firefox [13] or realplay). If the animation is stopped with the middle mouse button while in the menu area, then the movie stops until it is released by pressing the middle mouse button again. See "movi" for the keyboard options. Further remarks in "How to change the format of the movie file".

7.13 Help

Starts the html help and displays this document. By default it works with the specified html-viewer Firefox [13] but this can be changed in the "cgx.h" file. The search-path for the documentation is also defined in the "cgx.h" file. Please make sure that the documentation is in the specified location or change the path in the "cgx.h" file and recompile the program after the object-files are deleted. The default location for the html help is .../CalculiX/cgx_X.X/doc/cgx and /CalculiX/ccx_X.X/doc/ccx for cgx and ccx respectively. The html files can be downloaded directly or compiled from the latex source for this function to work properly. The INSTALL file explains how to compile the latex code to html. The INSTALL file is located .../CalculiX/cgx_X.X/ and .../CalculiX/ccx_X.X/ for cgx and ccx respectively.

If no compiler is available the user may use the "asgn" command to redefine the viewer and the help files (see also "Customization").

7.14 Toggle CommandLine

Add the command line from the konsole to the graphic's window and back. The user may consider to place the command "view cl" to a file named ".cgx" the user's home directoy for an automatic switch at startup.

7.15 Quit

This terminates the program without a save.

8 Customization

The file ".cgx" located in the $HOME directory will be read at program start. The following commands might be useful in this context:

- "asgn"
- "cmap"
- "col"
- "font"
- "menu"
- "rot"
- "subm"
- "view"
- "wpos"
- "wsize"

Example content of a ".cgx" init file:

```
cmap jet
menu readinp read /home/user/cgx_templates/readinp.fbl
asgn viewer xview
asgn viewformat png
asgn ccxhelpfile /usr/local/ccx.pdf
asgn cgxhelpfile /usr/local/cgx.pdf
asgn browser okular
```

9 Commands

This section is a reference to all commands and their parameters in alphabetic order. If a command is typed the mouse-pointer must be in the main window (figure 2). Only the echo of the input stream is visible in the parent konsole. The keywords are not case sensitive but all command parameters are case sensitive. Each reference starts with a short description of the command. The following syntax is used for these descriptions:

```
Known commands and syntax:
 '..':   Keyword (either uppercase or lowercase)
 <..>:   Parameter (case-sensitive)
 [..]:   combination of parameters or optional parameter
 (..):   Remark
 |   :   OR
```

```
&    :   AND
-    :   from-to
->   :   command continues in the next line
RETURN   press the RETURN key
```

Entities—with the exception of nodes and elements—are referenced by names which can contain letters and numbers. Usually one to four characters is recommended. If a new entity uses an existing name, the old definition will be overwritten. To overcome this problem, "alias" names can be used. An alias name is defined with the ! sign in front. An already defined alias name can be referenced by placing the % sign in front. For example:

LINE !L1 %P1 %P2 %SET

will create a line with the alias name L1 and will use the alias names P1 and P2 to define the end-points and uses the set SET to define the point sequence between the end-points. The assigned alias name for a given entity can be enquired with a leading question mark using the prnt command.

9.1 anim

```
'anim'   'tune' <value>|'steps' <value>|'time' <value>|->
         'real' ['on'|'off']|'model' ['on'|'off']| ->
         'elem' ['on'|'off']| 'start'
```

This keyword is used to manipulate the animation of displacements. See also "ds" and "scal". The amplification is controlled with "tune". "steps" defines the number of frames over one periode. "time" controlls the duration of one periode. "real" switches of the automatic amplification and the real displacements are used instead. In addition the displacements of the negative part of the periode is set to zero. "model" switches the static model (undeformed) edges on or of. "elem" does this for the element edges. Start the animation with 'start'. The animation stops when using the 'ds' or the 'view' commands with appropriate parameters.

9.2 area

```
'area' <set>
```

This keyword is used to calculate the area and the center of gravity of a set of shell-elements or surfaces of volume-elements. If a 'dataset' is active then an averaged value is calculated.

It averages the nodal values per element and weight it (multipies it) with the area of that element. The sum of all the weighted element-values is then divided by the total area of all regarded elements. The center of gravity is also weighted with the indvidual areas. This works for faces as well. In case the "qcut" command was used to create a section it is necessary to use the "comp"

command to add the related faces to the set '-qcut' which holds the section:

comp -qcut do

Then the 'area' command can be used:

area -qcut

which produces a listing like that:

```
AREA:98.437740  CENTER OF GRAVITY: 9.214960 0.663785
 24.655288 AVERAGE-VALUE:252.453576
```

The command writes to the "stack".

9.3 asgn

```
'asgn' 'n'|'e'|'p'|'l'|'c'|'s'|'b'|'S'|'L'|'se'|->
       'sh'|'alpha'|'beta'|'nadapt' <value>| ->
       'bg' 'on'|'off' ->
       'graph' 'on'|'off'|<nr>| ->
       'max'|'maxr'|'maxc' [<col>]|'minc' [<col>] ->
       'mem' 'free|'keep'| ->
       'netgen'|'tetgen'| ->
       'thrds' <value>|'rbe' [<value>|'mpc']| ->
       'usr' <text>| ->
       'viewer' <[path]file>| ->
       'viewformat' 'ps'|'png'| ->
       'browser' <[path]file>| ->
       'cgxhelpfile' <[path]file>| ->
       'ccxhelpfile' <[path]file>
```

This keyword is used to manipulate the behaviour of successive commands or certain values.

It can be used to define the first node or element number which will be used for the next mesh generation. And it is used to redefine the leading character of new entities. The default is D for points p, L for lines l, C for combined lines (lcmb) c, A for surfaces s, B for Bodies b, Q for nurb lines (nurl) L, N for nurb surfaces (nurs) S, A for sets se and H for Shapes sh. For example

asgn p U

will assign the character U as the leading character to all newly created names of points. The automatically created names of geometric entities use 4 characters. If all possible names with the chosen leading letter are in use then the next alphabetical letter is chosen as a leading letter, so after PZZZ follows Q000. After the last letter the amount of characters per name is increased. The maximum

number is 8. Each entity has its own name space. Different entities might use the same name. Remark: Currently nurbs-lines are automatically create splines sharing the same name. Nurbs-lines can not be used for other purposes than to be displayed and so far they can not be written to a file.

Meta informations stored in the frd database can be extended:

asgn usr ADDDISP DS:1

will assign or change the User Header Record "ADDDISP DS:1". Actually this type of record is automatically generated if the user adds displacements to the node coordinates (see "Toggle Add-Displacement)" and "User Header Record)"). The first word in the text is used as a key and the rest is the message. So

asgn usr ADDDISP NONE

will change this User Header Record to the new message "NONE".

The node related data will be readed if the user selects a certain dataset. The already stored data are kept by default. With the command

asgn mem free

the current data are freed before the selected data are stored. With

asgn mem keep

the current data are kept. This is useful on computers with limited memory.

The method how the the color fringe plot scale is presented can be chosen with either

asgn max

asgn maxr

or

asgn maxc m

and

asgn minc m

were 'm' is the color of the suppressed values, here magenta (see maxc).

The command is also used to control the behaviour of the surface mesh generator for unstructured triangles. This mesher [15] uses the tree parameters alpha, beta, nadapt for mesh-control. Current default is 0.4 for alpha and beta

and 4 for nadapt.

The surface mesh generator is able to use multiple cores (threads). Per default the mesher uses only one thread because only in this case the mesh numbering is reproducable. But since the surface meshing of a CAD geometry can be very time consumable a certain number of threads is used when the model is opened in the auto mode (cgx -a file). The actual number of threads is listed with the command "prnt info". This value can be changed in the cgx.h file (NTHREADS). When using

asgn thrds 8

the next mesh generation will use 8 threads and when saving the geometry this command is stored as well. The user might use this command to store his personal number of threads in the ".cgx" file in his home directory.

Currently two different tet mesher are available [4] [5]. They can be chosen with

asgn tetgen

or

asgn netgen

In case Nastran input should be generated it is possible to switch from MPCs to RBEs when using the send command in combination with the areampc option. The value after "rbe" represents the thermal expansion coefficient of this elements:

asgn rbe 0.5e-6

It should be noted that coincident nodes are connected by MPCs either way.

The presentation of the 2D plots generated by "graph" can be switched of or on and the next running number of the files written with 'graph' can be defined:

asgn graph off

asgn graph on

asgn graph 11

The viewer used for that presentation can be redefined with

asgn viewer /usr/bin/ghostview

which switches here to ghostview for ps files. The file type used for the 2D plots can be changed to 'png' with:

asgn viewformat png

But of course the viewer has to be changed as well:

asgn viewer xview

The helpsystem can be redefined with

asgn browser /usr/bin/okular

which switches here to an pdf viewer. Of course the helpfiles must now be redefined to pdf files with

asgn cgxhelpfile /home/user/cgx.pdf

and

asgn ccxhelpfile /home/user/ccx.pdf

Finally to speed up the execution of batch files in normal operation (with the grapical window visible) the internal operations necessary for graphical output can be switched off or on during reading of command lines from a file with:

asgn bg off

asgn bg on

9.4 aver

```
'aver' <set>
```

This keyword is used to average locations of nodes and/or points and of values at nodes.

9.5 bia

```
'bia'  <line>|<set> [ [<bias>] [<factor>]|
                ['mult'|'div' <factor>] ]
```

This keyword is used to define the bias of a single line or of a set of lines (see qadd). The bias defines the ratio of the length of the first element to the length

of the last element. For example,

bia all 4.5

will force a ratio in which the last element is 4.5 times bigger than the first one. Real numbers are permitted since version 1.5 (see also qbia). To convert from pre 1.5 versions, start the program with the -oldbias option. A negative factor permits to invert the direction of the bias:

bia all 4.5 -1

9.6 body

```
'body' <name(char<9)>|'!' [<surf1> <surf2>]|
            [<surf1> <surf2> <surf3> <surf4> ->
            <surf5> [<surf6>  <surf6>]]|
            [<set>]
```

This keyword is used to define or redefine a volume (body). Each body must have five, six or seven surfaces to be mesh-able with hexaeder-elements, otherwise it can only meshed with tets if NETGEN [4] is installed. However, it is sufficient to specify just the "top" and the "bottom" surfaces. But if surfaces with 3 or 5 edges are involved then this surfaces have to be the "top" and "bottom" surfaces. This is also true if surfaces have different line-divisions at opposite edges. The missing surfaces between the "top" and "bottom" surfaces will be created automatically if they do not already exist (they will always have 4 edges with the same division on opposide edges). But all needed lines must exist. More precisely, only single lines or existing combined lines (lcmb) can be detected. The user must define the missing surface if just a chain of lines (and no lcmb) is defined between two corner points of the "top" and "bottom" surfaces before he can successfully use the body command. It is a more convenient way to define a body than the command "gbod" but exactly 2 or all surfaces must be specified otherwise the body will not be created (The most convenient way to define bodies is to use the command "qbod"). For example,

body b1 s1 s2

will look for the missing surfaces and if necessary create them if all lines between the corner points of s1 and s2 are defined; the result is the creation of body, b1. Or for example,

body ! s1 s2 s3 s4 s5

will create a body and a new name for it. The new name is triggered by the

sign !. Here the body is based on 5 surfaces. If the surfaces are not connected, the body is not mesh-able.

In case a body should only be meshable with tets it can be composed of more than 7 surfaces. The definition can be provided by a set of surfaces:

body ! surfset

will create a body based on the surfaces referenced by surfset.

9.7 break

```
'break'
```

This keyword is used to end the interpretation of a command file. The program returns to the interactive mode.

9.8 call

```
'call' <parameters>
```

This keyword is used to allow the user to control his own functionality which he had coded in the file "userFunction.c". The default function "hydro" calculates the hydrodynamic stresses:

call hydro

To list all existing functions type:

call

See "User-Functions" for further details.

9.9 capt

```
'capt' <string>
```

This keyword is used to define the caption. This commend will show up in the menu area of the main window below the picture. Initially the filename is used as caption. A second line can be generated with "ulin".

9.10 cmap

```
'cmap' 'classic'|'gray'|'inferno'|'jet'|'coolwarm'|
       'turbo'|'viridis'
```

Changes the colormap of the color scale. The default is 'classic'. The user might use this command to store his personal setting in the ".cgx" file in his home directory. See also "steps" to adjust the nr of boxes.

9.11 cntr

```
'cntr' <pnt|nod|set>|[x y z]
```

Defines a new rotation center. See also "qcnt" for the cursor controlled command.

9.12 col

```
'col' <name> <red> <green> <blue> [<maxvalue>]
```

User defined new colors which can be used with plot/plus. Each of the red, green and blue values must be fraction of maxvalue (def:1). See also "prnt" how to list the available colors. The user might use this command to store his personal colors in the ".cgx" file in his home directory. For example:

col dg 3 89 26 155

define a dark green (dg) color.

9.13 comp

```
'comp' <set|*chars*> 'c'|'d'|'e'|'f'|'p'|'u'
```

This keyword is used to add all entities to the specified set (see seta) which depend on the already included entities (u, up), or to include all entities necessary to describe the already included entities (d, down).
For example the set "lines" stores lines and should also include all dependent points:

comp lines do

Or the set "lines" should also include all surfs and bodies which depend on the lines:

comp lines up

In some cases you will need only the end-points of lines. With the option p (edges)

comp lines p

only end-points and their lines are included in the set. Option c (combined lines) adds all LCMB's which use the stored lines. One exception to this logic was introduced for convenience:

comp nodes do

will add all faces which are fully described by the nodes in the set despite the fact that faces are made from nodes. For this purpose it uses the specific function were related faces can be added when using a set with nodes and/or elements with:

comp set f

As mentioned above they must be fully described by the nodes in set. With

comp set e

all elements which are fully described by the nodes in the set will be added.

Wildcards (*) can be used to search for setnames of a certain expression:

comp E* do

will complete all sets starting with "E".

9.14 cont

```
'cont'
```

This keyword is used to re-start reading of command files (see "stop").

9.15 copy

```
'copy' <set> <set> ['scal' <fx> <fy> <fz> <pnt> [a] ]|
                   ['tra' <dx> <dy> <dz> [a]]|
                   ['rot' <p1> <p2> <alfa> [a] ]|
                   ['rot' 'x'|'y'|'z' <alfa> [a] ]|
                   ['rot' <p1> 'x'|'y'|'z' <alfa> [a] ]|
                   ['rad' <p1> <p2> <dr> [a] ]|
                   ['rad' 'x'|'y'|'z'|'p'<PNT> <dr> [a] ]|
                   ['rad' <p1> 'x'|'y'|'z' <dr> [a] ]|
                   ['nor' <dr> [a] ]|
                   ['mir' <P1> <P2> [a] ]|
                   ['mir' 'x'|'y'|'z' [a] ]|
                   ['mir' <P1> 'x'|'y'|'z' [a] ]
```

This keyword is used to create a copy of a set (see seta about sets). Geometry, nodes and elements with their results can be copied. The copy of results is useful to evaluate additional sectors in case of a cyclic symmetric calculation. The copy is included in the new set. Existing sets are extended by the copied entities if the last parameter "a" (append) is provided. Several transformations are available. For example scal creates a scaled copy, the scaling factors fx, fy,

fz can be chosen independently,

Several transformations are available. For example scal creates a scaled copy, the scaling factors fx, fy, fz can be chosen independently,

copy part1 part2 scal 2 P0
copy part1 part2 scal 1 1 2 P0

tra will create a copy and will move it away by the vector dx, dy, dz and the optional parameter a will assign the new entities to sets were the mother of each entity is included (see "setl" on how to lock and therefore exclude certain sets from that behaviour),

copy set1 set2 tra 10 20 30 a

rot will create a copy and will move it around the axis defined by the points p1 and p2 by alfa degrees,

copy set1 set2 rot p0 px 20.

or the axis of rotation is given by specifying one of the basis coordinate axes: copy set1 set2 rot x 20.

or just one point and a vector of rotation is given by specifying one of the basis coordinate axes: copy set1 set2 rot p1 x 20.

rad will create a copy and uses the same transformation options as 'rot' or will create a spherical section if just a single point is defined,

copy sphere1 sphere2 rad pP0 10.

nor will create a copy and will move it away in the direction of averaged normal local vector. This requires information about the normal direction for each entity. Nodes will use associated element faces and geometric entities will use the element faces, surfaces or shapes which are stored with them in the set1,

copy set1 set2 nor 1.2 a

mir will create a mirrored copy. The mirror-plane is placed normal to the direction running from P1 to P2 and placed at P2,

copy section1 section2 mir P1 P2.

as with 'rot and 'rad' additional transformation options are available:

copy section1 section2 mir P1 x

places the mirror at P1 with its normal direction in 'x' direction

copy section1 section2 mir x

Places the mirror in the origin with its normal direction in 'x' direction.

9.16 corrad

```
'corrad' <set>
```

This is a very special command to adjust improperly defined arc-lines, like in fillets. The center points of arc-lines included in the set are moved in a way that each arc-line will run tangentially into a connected straight line. But because the end-points of the arc-lines are not moved only one side of each arc-line will run into a connected line. The other side is not controlled and might end in a sharp corner. Therefore for each arc-line exactly one connected straight line must be included into the set (figure 4).

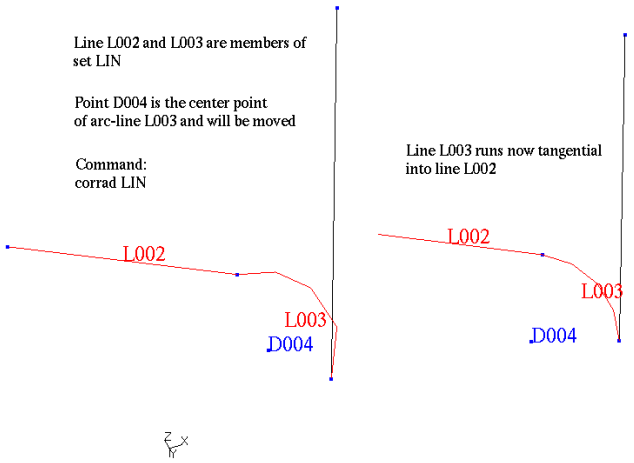


Figure 4: Effect of the corrad command

9.17 csysa

```
'csysa' <sysNr> <set>
```

Specifies the displacement coordinate system for each node (Nastran only).

9.18 cut

```
'cut' [<set>|<nod>] | [<pnt|nod> <pnt|nod> <pnt|nod>]
```

This keyword is used to define a cutting plane through elements to visualize internal results. New elements are generated and stored in the set "-qcut" for this purpose. The plane is either defined by three nodes or points, or by just one node if a vector dataset-entity was already selected. The cutting plane is then determined by the direction of the vector (displacements, worstPS, ..). The menu option "Show Elements With Light" or the commands "ucut", "view surf" or"view volu" will display the whole model again and will delete the elements generated by cut or qcut. This command is intended for batch-mode. See "qcut" for the cursor controlled command. The three nodes or points may be provided by specifying a set of either unordered or ordered type (a sequence). The elements generated by successive qcut or cut add up until they are deleted.

9.19 del

```
'del' ['p'|'l'|'c'|'s'|'b'|'t'|'v'|'S'|'L'|'se'|'sh' <entity>]|
      ['l0' <entity>]|
      ['se0']|
      ['mesh']|
      ['pic']
```

This keyword is used to delete entities, the whole mesh (see also qdel) or a background-picture. For example,

del se part

will delete the set "part" but all included entities are untouched. The following entities are known:

Points p, Lines l, Combined Lines c, Surfaces s, Bodies b, Node Texts t, values v, Nurb Surfaces S, Nurb Lines L, Sets se and Shapes sh.

When an entity is deleted, all dependent higher entities are deleted as well. Special cases are

del l0 set (l¡zero¿)

were all lines with zero length in set "set" are deleted and

del se0

will delete all empty sets. If a value is to be deleted it is necessary to mask it with a leading '\' otherwhise it would be replaced by its content:

del v \val

will prevent the substitution of val1 by its content.

If a background-picture was loaded with the "read" command it can be deleted with:

del pic

See also "zap" on how to delete a set with all its referenced entities.

9.20 dist

```
'dist' <set> [<target-set>|<shpe>] ->
           ['tra' <dx> <dy> <dz> <offset>]|
           ['rot' <p1> <p2> <offset>]|
           ['rot' 'x'|'y'|'z' <offset>]|
           ['rad' <p1> <p2> <offset>]|
           ['rad' 'x'|'y'|'z' <offset>]|
           ['nor' <offset> <tol>]
```

measures distances between entities of one or of two sets. For example

dist set

gives the distance range of the points and nodes in the set in x,y and z direction.

If just one point or node is included in both provided sets

dist set1 set2

then as with "qdis" the cartesian distances, the angular distances and the radial distances are determined.

If just one point or node is included in the first set and several points or nodes in set2

dist set1 set2

then only the extremal cartesian distances from the sole point or node in set1 to all entities in set2 are determined (set1-set2).

Further parameter combinations work analogous to the "proj" command and calculates distances to shapes or surfaces. Please look there for details, for example:

dist set1 set2 rot p0 px

The average-, maximum- and minimum distance is determined. The distance is measured normal-, rotational-, radial or translatoric.

The command writes to the "stack".

9.21 div

```
'div'  |
       <defdiv>|
       <line> [<division>]|
       <set>  [<division>]|
              ['mult'|'div' <factor-div> <factor-bias>]|
              ['auto' <node-dist> <angle> <elem-ratio>]
```

This keyword can be used to re-define the default division of lines:

div 4

The div keyword works on a line or a set of lines (see qadd). The division controls the number of nodes created when the geometry is meshed (see elty and mesh). For example,

div all 4

attaches the division of 4 to all lines. With the keyword mult or div in combination with a value, it is possible to multiply or divide already assigned divisions:

div all mult 2.

Or in case you need a starting-point for the individual divisions you can use the option auto with the optional parameters node-dist and angle. Node-dist is the maximum allowed distance between nodes and angle is the maximum allowed angle defined by three sequential nodes. If one parameter is not fulfilled then the division is halved until the requirements are fulfilled. Default values are defined in the file cgx.h and can be listed with

div

without parameters

div all auto

uses the defaults. The following example sets them expicidly:

div all auto 2. 10. 0.5

will use a maximum element lenght of 2., the angle between successive nodes

is less than 10 degree and the minimum element is only half of the maximum-length as long as the length of the line is sufficient. It should be noted that it could make sense to use different values for different sets.

9.22 ds

```
'ds' [<1.Dataset-Nr> [<2.Dataset-Nr>] [<n.Dataset-Nr>] ->
         ['a[h]'|'e[h]' [<entity-nr> (up to 4 times)]]|
         [['+'|'-'|'*'|'/'] <Dataset-Nr> ['c'] ]|
         ['o' <value> [<entity-nr>]]|
         ['p' <power> [<entity-nr>]]|
         ['s' <value> [<entity-nr>]]|
         ['r' <key> [<parm1>] [<parm2>] [..<parm5>]]]|
     ['g' <name> [[<ncomps>|<0>] <value> <text> <type> ->
       <step> <analysisName>]]|
     ['e' <name> <comp> <type> <row> <column>]|
     ['f']
```

This keyword is used to select, modify or generate one or more Datasets (ds) and one or more Entity (e). In addition it can be used to generate or modify related parameters which might store step specific descriptions. The dataset has to be a positive number which has to match the nr in the Dataset-menu or an 'l' (lower case 'L') which is interpreted as the last available Dataset or a negative number. Then it is interpreted as the last ds minus the specified number. For example

ds 1

will just select the first Dataset and will write some informations about it to stdout and to the "stack" if active. It will write ds-nr, lc-name, ds-value, ds-text, ds-name, nr-of-entities

ds 1 e 1

will display the first entity of the first Dataset and will write some informations about it to stdout or to the "stack" if active. It will write ds-nr, lc-name, ds-value, ds-text, ds-name, entity-name.

ds l e 1

will display the last Dataset. To start the animation of the second-to-last Dataset right away:

ds -1 a

Or generate an animated fringe plot by adding the desired entity:

ds -1 a 4

Sequences can be defined by specifying one to three datasets and by extending the 'e' parameter by an 'h' ('history'):

ds 2 eh 1

Here all datasets of the same type as ds 2 are selected. The spacing between datasets of the same type is only evaluated for the first step. A unique step-with is therfore needed.

ds 2 10 eh 1

Here the datasets 2, 10 and all successive ones of the same type with a spacing of 8 are selected.

ds 2 4 10 eh 1

Here the 1st entity of each second Dataset is selected. The selection starts at the second- and ends at the 10th dataset. If more than one entity is defined then a vector-plot will be displayed. If a 4th entity is defined then this entity will be used for the basic color-plot:

ds 2 4 10 eh 12 13 14 15

In case the deformed shape should be shown together with the fringe plot in a sequence of datasets then the 'e' parameter has to be replaced by an 'a' character.

ds 2 ah 1

REMARK: So far vector plots can not use the deformed shape. Therefore only one entity is supported.

In addition, it is possible to scale or offset the entities of the specified datasets:

ds 1 s 1.2

will scale all entities of dataset 1 by a factor of 1.2.

ds 1 + 3

will create a new dataset which stores the addition of dataset 1 and 2.

ds 1 + 3 c

using parameter 'c' will change the dataset 1 with the result of the addition of dataset 1 and 2.

ds 1 p 1.2 3

will use the given exponent to scale entity 3 of dataset 1 by an exponent of 1.2.

ds 1 o 200.

will add a value of 200 to all entities of dataset 1.

ds 1 o 200. 2

will add a value of 200 to the entity 2 of dataset 1. Each dataset might use related parameters (see Parameter Header Record for the format of a parameter record). This parameters can be overwritten or created:

ds 2 4 10 r TAMB 1

Each second dataset from 2 to 10 gets a related parameter 'TAMB' with the value '1'. If the value is nummeric it can be used by the "graph" command.

A new dataset in which all values are initiallized to zero is generated with:

ds g VELOCITY 3

The 'name' VELOCITY will appear in the menu as the dataset name and can be 8 character long. It has 3 components ('ncomps', default is '1'). The other parameters are optional:

- value: A nummeric value, usually time or frequency (used by "graph").
- text: A describing text (used by "graph").
- type: Analysis type (static:0,time step:1,frequency:2, etc.).
- step: Step or increment number
- analysisName: Type of analysis (description).

The current dataset name is modified if only the name is given:

ds g VELOCITY

The other parameters of the current dataset can be modified if the number of components is set to zero: ds g VELOCITY 0 1e4 test

The entities of the current dataset are manipulated with:

ds e V 2

The 'name' V will appear in the menu as the entity name and can be 8 character long. It is the second entity (component) of the current dataset (default is '1'). The third and following parameters can be omitted for a scalar.

- name: entity name
- comp: entity nr (component)
- type: Mathematical data type (scalar:1, vector:2, matrix:4, etc.)
- row: sub-component index or row number
- column: column number if matrix

The 'row' paramter defines the location of the enitity in the vector or the row of the matrix. The dataset is finalized with:

ds f

The maximum- and minimum value over all nodes for the current dataset will be determined and stored together with the corresponding node numbers. This values are needed for the graphic display.

The values at the nodes are manipulated with the "node" command. With that command the new dataset gets its data. See also "How to generate a new dataset" for further details.

More details can be found in section "Nodal Results Block".

9.23 elem

```
'elem' <nr|!> [set]|
    [<firstNode> .. <lastNode> 'be2'|'be3'|'tr3'|'tr6'|->
    'qu4'|'qu8'|'he8'|'he20']
```

This keyword is used to define elements based on nodes and its type (see section Element Types in the appendix for the correct node-order). For example,

elem 1 1 2 3 4 qu4

creates a shell element with four nodes. If the an automatically generated name is desired, then type "!" instead of a name. Shell elements can be created based on a set of element-faces:

elem ! faceset

This might be useful to create a layer of shell elements on volume elements.

9.24 else

```
'else'
```

See the "if" command.

9.25 else if

```
'else if' <value> 'eq'|'ne'|'=='|'!='|'<'|'>' <value>
```

See the "if" command.

9.26 elty

```
'elty' <set> 'lock'|'ulock'|->
             'be2'|'be2r'|'be2f'|'be2d'|->
             'be3'|'be3r'|'be3f'|->
             'tr3'|'tr3u'|'tr3e'|'tr3s'|'tr3c'->
             'tr6'|'tr6u'|'tr6e'|'tr6s'|'tr6c'|->
             'qu4'|'qu4e'|'qu4s'|'qu4c'|->
             'qu4r'|'qu4er'|'qu4sr'|'qu4cr'|->
             'qu8'|'qu8e'|'qu8s'|'qu8c'|->
             'qu8r'|'qu8er'|'qu8sr'|'qu8cr'|->
             'he8'|'he8f'|'he8i'|'he8r'|->
             'he20'|'he20r'|'pe6'|'pe6f'|'pe15'|->
             'pe15r'|'te4'|'te4f'|'te10'|'te10m'|->
             'te10t' [<parameter>]
```

This keyword is used to assign a specific element type to a set of entities (see section Element Types in the appendix). In most cases it can be used to specify the element type before the mesh is created. In case of unstructured meshes more specific element attributes have to be assigned after the mesh was created (from tr6u to tr6c or te10 to te10t etc.).

The element name is composed of the following parts: The leading two letters define the shape (be: beam, tr: triangle, qu: quadrangle, he: hexahedra, pe: penta, te:tetraeder), then the number of nodes and at last an attribute describing the mathematical formulation or other features (c: axisymmetric, e: plain strain, s: plain stress, u: unstructured mesh, r: reduced integration, i: incompatible modes, f: fluid element for ccx, t: initial temperatures are interpolated linearly within the tet element (ccx:C3D10T)).

If the element type is omitted, the assignment is deleted. If all parameters are omitted, the actual assignments are posted:

elty

will print only the sets with assigned elements. Multiple definitions are possible. For example,

elty all

deletes all element definitions. If the geometry was already meshed, the mesh will NOT be deleted. If the mesh command is executed again after new assignments have taken place, additional elements could be created.

elty all he20

assigns 20 node brick-elements to all bodies in the set all.

elty part1 he8

redefines that definition for all bodies in the set part1.

elty part2 tr6u

assigns 6 node unstructured triangle elements to all surfaces in set part2.

elty part2 tr6u 0.5

will do the same but specifies a mesh refinement factor of 0.5 (>1: coarser than the average boundary spacing, <1: denser). Be aware that specialized unstructured meshes must be created by using two times the elty command. First time the general unstructured type before the mesh is actually created and afterwards a redefinition into the more specific type:

elty part2 tr6u
mesh all
elty part2 tr6c

creates an axisymmetric unstructured mesh.

elty part3 te10

assigns 10 node elements to all bodies in set part3. But this works only if TETGEN [5] or NETGEN [4] is installed and the locations of their binaries tetgen or ng_vol are included in the path system variable.

elty part3 te10 3.5

will do the same but specifies a target size for the elements. In this case the modified program ng_vol from the cgx-distribution must be available. Replace the original ng_vol in the NETGEN package and build it again. Be aware that specialized unstructured meshes must be created by using two times the elty command. First time the general unstructured type before the mesh is actually

created and afterwards a redefinition into the more specific type:

elty part2 te10
mesh all
elty part2 te10t

The penta element types are not supported for meshing but elty can be used to redefine the attributes (pe6 to pe6f). Penta elements are only created if a mesh of triangles (2D) is sweeped in 3D. This procedure is used to create quasi 2D cfd meshes.

Sometimes it is necessary to prevent element definitions from beeing changed again. This can be achived with

elty ¡set¿ lock

the element definition and mesh density of all referenced bodies, surfaces and lines is fixed until unlocked with:

elty ¡set¿ unlock

The fluid network element types are be2f and be3f. The be2f has to be used at the begin and the end of a network. This elements will use the special node nr '0' at the entry and exit. This node-nr is automatically assigned to the element definition when written in the ccx format. All other elements in the network must use the be3f type. The detailed element type definition has to be done in the ccx input file based on element-sets.

9.27 endif

```
'endif'
```

See the "if" command.

9.28 endwhile

```
'endwhile'
```

See the "while" command.

9.29 enq

```
'enq' <set> <set> ['set' <setname> ]|->
          ['rec' <x-value>|'_' <y-value>|'_' <z-value>|'_' ]|->
          ['cx'|'cy'|'cz' <r-value>|'_' <fi-value(deg)>|'_' ->
            <axis-value>|'_' ] ->
```

```
<tol> 'i'|'a'|'h'|'l' [<value>]
```

This command is used to locate entities from a certain set (first provided set) and stores them in the second set. The following entities are identified: nodes, points, lines, surfaces. But surfaces can only be identified if the command "rep <setname>" was issued before. In case of nodes with related values (results) it will also determine the highes or lowest value in the specified range, or, all nodes above or below a certain value. The coordinate of the search location might be taken from one node or point in a given 3rd set or given in cartesian coordinates (option rec) or cylindrical coordinates (option cx, cy, cz). Some coordinates might be omitted to specify an infinite range. The '_' key has to be used in this case. The mode is defined by the keys 'i' individual, 'a' all, 'h' high, 'l' low, were h and l will search the highest or lowest value in range. This value will also be written to a file for automatic processing purposes. For example

enq all newset rec 10. _ 100. 0.1 i

will search for entities in set 'all' at the given location x:10. y:_(infinite range, triggered by the '_' key) z:100. with a tolerance of 0.1 and only the closest entity of all kinds of entities are stored in set 'newset', triggered by the 'i' key. With the 'a' key all in range would be stored. The key 'h' or 'l' would trigger a search for the highest or lowest value in the specified range. If the search should be done close to a certain node:

enq all newset set nodeset 0.1 h

will search for the node with the highest value around the node stored in nodeset inside a radius of 0.1.

In combination with a certain value

enq all newset rec 10. _ 100. 0.1 h 1013.

all nodes with a value above "1013." would be stored in set "newset". The command

enq domain1 newset cx 100. 180. 10. 0.1 a

will search for entities in set 'domain1' at radius:100. around x at fi:180 degree and x:10. with a tolerance of 0.1. All entities in range are stored in set 'newset'.

The following example illustrates the use of "enq" to find the highest value close to a location given by a node-number:

```
# activate dataset 3 with entity 7:
ds 3 e 7
# read a file with one node to define a set:
```

```
read pos1.frd pos1
# search the highest value around a radius of 1:
enq all t set pos1 1. h
# move the file with the search result to a meaningfull name:
sys mv enq_lc3_e7_1.out pos1.out
```

One important remark: The command will select not only nodes in the selection range but also geometric entities (lines etc.) which in turn reference their related nodes. This behaviour can mix up the desired result when the user expects to find just nodes in the selection range. Therefore it is better to first generate a set only with nodes as a basis for 'enq':

seta nodeset n all
enq nodeset new ...

The command writes to the "stack".

9.30 eprop

```
'eprop' <set>
```

Calculates the maximum element edge length, volume and quality factor. The maximum values are stored at nodes in a new dataset. Not all element types are supported. Second order elements are treatened as linear regarding length and quality. The quality measure used is proportional to the ratio of the longest edge divided by the radius of the inscribed sphere. The proportionality constant is such that the quality is 1 for an equilateral tetrahedron. For all other elements it exceeds 1. The bigger this number, the worse the quality. See also "eqal".

9.31 eqal

```
'eqal' 'jbir'|'aspr'|'mca' <value>
```

The keyword without parameters lists the current element quality thresholds (0==off):

- JBIR: The maximum ratio of jacobian determinants xsj between all gauspoints. Were xsj = Vworldcoordinates / Velementcoordinates and jbir = xsj-max / xsj-min. Supported element types he8, he20, te10.
- ASPR: Element aspect ratio between all element sides. Supported element types he8, he20.
- MCA: Maximum corner angle between all element sides. Supported element types he8, (he20 not regarding midside nodes).

Parameter "value" sets the value of the threshold. The element-quality is checked with "plot eq all". See also "eprop". The user might use this command to store his personal parameters in the ".cgx" file in his home directory.

The command writes to the "stack".

9.32 exit

```
'exit'
```

This command terminates the program and saves the geometry (if any) to a file named as the input file but with the extension .fbd. If a file with that name exists already, then this file will be saved with the new file extension .fbb as a backup (see also save).

9.33 fil

```
'fil' <line> <line> <radius>
```

This command creates an arc of a given radius between the two specified lines.

9.34 flip

```
'flip' [<set> <e>|<b>|<s>]|
       [<s> ['auto']]
```

This command changes the orientation of a set of shell-elements, bodies or surfaces. In case of a single surface with the parameter "auto" all surfaces will be flipped in a way that they share a common direction (in or out of a volume). The related elements are also flipped. See qflp for the cursor controlled command.

9.35 flpc

```
'flpc'
```

This command changes the colour of the scale. Initialy the default is red for high values and blue for low values. The command will invert the current state.

9.36 font

```
'font' 'd'|'l'|'c' <value(1-6)>
```

This command changes the drawing-font. Six fonts of different heigh are available:

font d 6

selects the greatest font for the drawing area were

font l 1

selects the smallest one for the legend. The default is selected if no number is specified. The font for the command line is selected with

font c 6

9.37 frame

```
'frame' [<setname>]
```

This command fits the model or the contents of a given set into the drawing space. This command is usually automatically triggered.

When executing commands which increase the used space then it might happen that geometry is clipped and can not be seen or accessed anymore.

9.38 gbod

```
'gbod' <name(char<9)>|'!' 'NORM' '+|-' <surf> '+|-' <surf> ->
.. ( 5-7 surfaces )
```

This keyword is used to define or redefine a body in the most basic way (see also "qbod"). Each body must have five to seven surfaces to be mesh-able. However, the number of recommended surfaces is six. The first two surfaces should be the "top" and the "bottom" surfaces. For example,

gbod B001 NORM - S001 + S002 - S005 - S004 - S003 - S006

will create a body B001. The keyword "NORM" is a necessary placeholder for future functionality but has no actual meaning. Next, follow the surfaces with a sign "+" or "-" in front that indicates the orientation of each surface. These signs will be corrected automatically if necessary. If the an automatically generated name is desired, then type "!" instead of a name.

9.39 gonly

```
'gonly' 'on'|'off'
```

This keyword is used to erase the contents of the menu area. Sometimes this is useful for hardcopies.

9.40 graph

```
'graph'  [<amplitude|*chars*> 'amp']|
         [<material|*chars*> 'mat']|
         [<set|seq> 'length' ['+'|'-'|'+c'|'-c']]|
         [<set>|'-p' 'nr'|'time'|'descr'|<parameter-name> ->
           [<Dataset-name> <entity-name>|<parameter-name> ] ->
             [<first-Dataset-Nr> <last-Dataset-Nr>]  ]
```

This keyword is used to create time history plots of nodal values. The values of the pre-selected nodes stored in the referenced set will be written to a file called "graph_<nr>.out". A gnuplot command file will be written called "graph_<nr>.gnu" and executed. The resulting file "graph_<nr>.ps" will be automatically displayed with the postscript viewer unless the user has used "asgn"

to turn the presentation off (and the <nr> can be defined). The default viewer is "ghostview" but this can be changed in the "cgx.h" file before the program is build. See also "How to generate a time-history plot" for further details.

A set with the nodes of interest must be created (usually with "qadd") before a graph can be drawn. If the parameter l (length) is used to plot values along the length of a set of nodes then this set should be of the sequence type (usually created with "qseq"):

graph seq length

The length is calculated as the sum of the distances between successive nodes. Usual sets (as created with "qadd") will be copied into a temporary ordered set in a way that the positive direction points away from the origin of the coordinate system. The direction can be chosen with a '+' or '-' sign following the 'length' parameter ('+' is default). This temporary ordered set is kept when a 'c' follows the sign. The name of this set is equal to the original setname but with an 'N' in front.

Instead using existing nodes it is possible to use a line or a combined line "lcmb" as a method to generate new locations for data-points. The current displayed results will be automatically mapped to this locations and shown in a 2D graph over the length of the line:

graph lineset length +

The set lineset stores a line. Since a '+' was given the graph starts at the beginning of the line. With a '-' it would start at the end of the line.

Since version 1.8, there are two ways of operation. One way is to specify the Datasets and the entity as parameters of the command

graph set time DISP D1

which will display the displacement D1 of nodes in "set" vs. the dataset-time of all "DISP" Datasets. The first and the last dataset can be specified optionally.

For the traditional way, a sequence of datasets must have been selected (see Toggle Dataset Sequence). After the selection of the datasets, an Entity must be specified. Then, the user could use the graph command to generate the history-plot of this pre-defined sequence. The command

graph set nr

will display the values on the node-positions vs. the dataset-numbers. The command

graph set time

will display the values on the node-positions vs. the dataset-values (usually time

or frequency, see Nodal Results Block, parameter "VALUE")) and the command

graph set descr

will display the values on the node-positions vs. the dataset-description (only makes sense if the description is a numerical value. See Nodal Results Block), parameter "TEXT", for the location of this data in the frd-file.

In addition, a second data-file is written "graph.out2" which stores the node-number and the node-coordinates and all values at this node-position for all datasets. This file can be used to display values over node-positions, etc. It is up to the user to generate a suitable plot out of this data.

The "graph" files can be edited and combined. For example to plot one type of value vs. another type of value.

If an ccx- or Abaqus-input-file was read then it is also possible to create time history plots of the amplitudes (*AMPLITUDE in ccx) or the material-properties can be displayed (*MATERIAL in ccx):

graph amplitude-name amp

graph material-name mat

Wildcards (*) can be used to search for amplitudes or materials of a certain expression.

The parameters related to datasets can also be displayed with the graph command:

graph -p time DISP HID

This command plots the nodal-diameters used in a cyclic symmetry calculation over the dataset-value. Only few parameters are written from ccx in frd-format. Other applications might define its own parameters in frd-format (see Parameter Header Record for the format of a parameter record). If needed cgx can also create this dataset parameters, see the "ds" command on how to do it.

The running number in the name of the written file is stored in the value !graph_Nr (the leading '!' prevents the value from beeing written by the send or save command). It can be used in a subsequent "sys" command to change the filename.

9.41 grpa

```
'grpa' <nr> <set> ['dyna']
```

This command allows to assign elements to a group. The group number is an element attribute. One element can only belong to one group. This command assigns a group number to certain elements stored in a given set. An ls-dyna *PART command is written to a file if the parameter dyna is given. The group number is used as a property ID in an eventually written mesh file in ls-dyna format. See “Remarks Concerning ls-dyna” and “Element Definition Block” how this applies to the result format (frd).

9.42 grps

```
'grps'
```

The result format (frd) allows to assign elements to a group. One element can only belong to one group. This command creates sets and stores all elements of a certain group in a certain set called “+grp[nr]”. See “Element Definition Block” how this applies to the result format (frd).

9.43 gsur

```
'gsur' <name(char<9)>|'!' '+|-' 'BLEND|<nurbs>|<shape>' ->
       '+|-' <line|lcmb> '+|-' <line|lcmb> .. (3-5 times)
```

This keyword is used to define or redefine a surface in the most basic way (see also qsur). Each surface which is intended for a regular mesh must have three to five edges which might consist of a single line or a combined line (see lcmb) to be mesh-able. However, the recommend amount of edges is four. For example,

gsur S004 + BLEND - L002 + L00E + L006 - L00C

will create the surface S004 with a mathematically positive orientation indicated by the "+" sign after the surface name. The keyword "BLEND" indicates that the interior of the surface will be defined according to Coons [14] or a NURBS surface (nurs) or shape (shpe) is referenced. It should be mentioned that only nurbs or shape related surfaces can be meshed with unstructured triangles. Use a "+" or "-" in front of the lines or lcmbs to indicate the orientation. These signs will be corrected automatically if necessary. If the surface is intended for an unstructured mesh it is not necessary to care about the correct orientation of the lines and the number of edges is not limmited. If automatic name generation is desired, then use "!" instead of a name.

9.44 gtol

```
'gtol' [RETURN]|<auto>|<geometric-tol>
```

This keyword is used to enquire the default geometric tolerance:

gtol

Or it is used to recalculate the geometric tolerance:

gtol auto

Or it is used to set the geometric tolerance:

gtol 0.1

The merg command will recognize points or nodes as equally located when the distance between them is less than gtol.

The command writes to the "stack".

9.45 hcpy

```
'hcpy' [['gif'|'png'|'ps'|'tga'] ['name']]|
       [make [ls]]|[clean]
```

This keyword creates a hardcopy in one of the above formats. Default is tga. Use the program "convert" to convert to other formats if needed.

hcpy ps

will create a ps file with the default name hcpy_<nr>.ps

hcpy ps name

will create name.ps

hcpy make ls

will bundle all ps files created in one file using the landscape (ls) format. The ls parameter may be omitted. A maximum of six pictures is placed on each page.

hcpy clean

will remove all "hcpy" files. See also the commands "ulin", "rot", "ds", "max", and "min" which might be used in hcpy batch files. These commands should be used in the shown order before using the "hcpy" command.

The command writes the file-name and -nr to the "stack".

9.46 help

```
'help'
```

This keyword prints a short overview of all commands.

9.47 if

```
'if' <value>|<const> 'eq'|'ne'|'=='|'!='|'<'|'>' <value>|<const>
```

This keyword is used to compare two values ("valu" or constant numbers). If the compare is True the following commands are executed. If the compare is False the code after 'else' is executed. Normal operation continues after 'endif':

if arg1 == arg2

will skip successive commands when the numerical value stored in 'arg1' is not equal to the numerical value stored in value 'arg2'. The values are locally converted to 'float' format for the numerical comparison. The 'eq' and 'ne' compare strings and should not be used for numerical values since no conversion to a common format is done. Two strings are equal if they have the same length and all characters are equal.

See also "while", "valu", "stack" and "How to run cgx in batch mode".

9.48 int

```
'int' <line> <line>
```

This command creates the intersection point between the two specified lines. The longer part of the given lines is kept. The shorter part will be erased.

9.49 init

```
'init' <internal parameters>
```

This command defines the window and model size, the model position, rotational center, fonts and others. It is created and written to a file by the "send" command. If that command should change the window dimensions and is used in a batch file, it has to be used in the first line. Otherwhise the window dimensions take only effect after the batch file was completely parsed and executed. A batch file could look like that::

```
read init.fbl
read ccx.frd new
ds 1 e 1
plot fv all
hcpy png
quit
```

If the window size is not affected by the command then it is not necessarily the first command in the batch file.

9.50 lcmb

```
'lcmb' <name(char<9)>|'!' ['+|-' <line> '+|-' <line> '+|-' ->
<line>..(up to 14 lines)]|
                        ['ADD' '+|-' <line> '+|-' <line> ->
'+|-' <line>..(up to 14 lines)]
```

This keyword is used to define, extend or redefine a combined line (lcmb). Combined lines are necessary if the edge of a surface should be made of more than one line. Usually the user does not create lcmb's directly. They are created automatically during the process of defining a surface with the command qsur. There is no limitation to the number of lines in a combined line. However with one command, not more than 14 lines can be specified at a time. To specify more than that or to extend an existing lcmb a modify command has to follow. For example,

lcmb U260 + U249 - U248 - U247 - U243 - U237 - U236 - U231 - U219

defines the lcmb U260 with 8 lines and their orientation in the lcmb. The following command

lcmb U260 ADD - U218 - U217

extends the lcmb U260 by two additional lines.

It should be noted that an existing lcmb can be converted into a spline with the command seqc.

9.51 length

```
'length' <set>
```

This keyword is used to calculate the length of all lines stored in a set.

The command writes to the "stack".

9.52 line

```
'line' <name(char<9)>|'!' <p1> <p2> <cp|seq> <div> [<bias>]
```

This keyword is used to define or redefine a line. A line depends on points. A line can only be defined if the necessary points are already defined. Attention: The points p1 and p2 must not lie on the same location. There are three different types of lines available. The straight line

line l1 p1 p2 4

is defined by: its name l1 (the name could have up to 8 characters), by the points p1 and p2 and optionally by the division. The arc

line ! p1 p2 cp 4

needs a center point cp. The radius changes linear from p1 to p2 if the center-point cp is excentric. The name is chosen automatically (triggered by the character !). The spline

line l1 p1 p2 seq 4

needs a so called sequential-set, seq (use the command "seqa" or "qseq" to define such a set). This set seq stores the spline points between the end-points in the right order. The spline function is described in [16]. Usually, a line is defined interactively with "qlin".

The bias is used to change the distances between successive nodes:

line l1 p1 p2 4 1.1

The bias of 1.1 means that each successive node is 1.1 times farther away. Only if a bias is given can the division be greater than 99.

ATTENTION: If the user does not provide a bias only the last two digits of the given 'division' are used to define the division itself. The leading digits define a bias. But in a different way as explained above. Here the bias defines the length increase between the first and last element on the line. If the bias uses more than one digit in front of the division, the number has to be divided by a factor of ten to get the bias (10130 means a bias of 10.1 and a div of 30). A negative number changes the direction of the bias:

line l1 p1 p2 -404

The number -404 specifies a division of 4 with a bias of 4 but the first node distance is 4 times longer than the last.

9.53 lnor

```
'lnor' <name|!> [<set>|[<p1> <p2> <p3>]] <length>
```

A new line normal to a plane defined by three points and a length is created. It starts at the last point. The three points may be provided by specifying a set of either unordered or ordered type (a sequence).

9.54 mata

```
'mata' <Material-Nr> <set>
```

This keyword is used to assign a material-number to a set of elements. Currently, this feature is only useful if the NASTRAN format is used. The material-number

is a numeric attribute assigned to each element of the mesh and will be stored with the mesh in the frd-format or nastran-format (see send). For example,

mata 7 part

assigns the material-number 7 to all elements in the set part. Elements can have just one material-number. The default number is 1. These numbers will be saved with the mesh if the database is written to the file-system with the command:

send all frd or send all nas

If the frd-file is used later, the material-number(s) are available immediately.

9.55 map

```
'map' <slave-set> <master-set>  ->
      [['surf']|
       ['volu']|
       [ 'x'|'y'|'z'|'rx'|'ry'|'rz']]  ->
       'ds'[<nr>] ['nex']
```

This keyword is used to map (or interpolate) values from one mesh to another. For example

map slave master surf ds1

will map the values from Dataset 1 to the nodes of set slave. All available datasets will be mapped if no number follows the "ds" parameter. The parameter "surf" is used for mapping of values from element faces to element faces and their nodes (2D or 3D). Do not use internal nodes or volume elements! A typical application would be the mapping of pressure. The parameter "volu" triggers the mapping from a 3D mesh to another 3D mesh of the same shape (i.e. temperatures). The parameters "x,y,z" are used for mapping from 2D to 2D/3D in the indicated direction. The parameters "rx,ry,rz" are used for rotational mapping of 2D to 2D/3D around the x,y,z-axis. The "master" set must include the nodes and their elements. The optional parameter 'nex' prevents extrapolation and outside slave nodes get a zero value and they are stored in set '-notMapped'. Unconnected nodes are not allowed. Usually the master nodes (on which the values are known) were included from an external result file with the "read" command (with the parameter "add"). For further details see "How to map loads" in the appendix. Remark: The 3D mapper uses multi threading. The number of cores is defined in cgx.h (NTHREADS).

9.56 mats

```
'mats'
```

This keyword is usually used when a mesh with mateial numbers was read. For example a native-netgen file [4] or an frd file (see "Element Definition Block"). It lists the used material numbers and will generate sets for each material. The solver ccx writes the original material names in the frd file after the related material number (see "User Header Record" and "prnt usr").

9.57 max

```
'max' <value> ['e'|'f'|'i'|'l'] ['l'|'u']
```

This keyword is used to define the upper value of the scale in the menu area (see figure 2). The number representation can be changed between exp, float, int or log10 for which the scale is restricted to positive values. For example

max 1100 i

will set the upper value to 1100 and the representation to integer. A third parameter "l" (lock) or "u" (unlock) can be provided which locks the scale to certain max or min values. The selection of a different dataset will not change the scale. The specified value defines the highest value of the last box. See also "maxr", "min".

9.58 maxc

```
'maxc' <value> ['e'|'f'|'i'|'l'] ['l'|'u']
```

The functionality of "maxr" but the color of the last box is 'n' (neutral) unless the user had changed this color by using the asgn command.

9.59 maxr

```
'maxr' <value> ['e'|'f'|'i'|'l'] ['l'|'u']
```

The functionality of "max" but the specified value defines the lowest value of the last box. See also "maxc"

9.60 menu

```
'menu' <name> <command>
```

This keyword is used to add a cgx command line to the menu. It will appear in the submenu 'user'. The command is usually a 'read' command which loads and executes a file with cgx commands:

menu checkbou read cb

were 'cb' is a value ("valu") which stores the path to a file to check boundary conditions. The file could have been used directly but using a value is convenient since it is then possible to start the same action with a quite short command line ("read cb") instead of typing the full file name or using the menu. This commands are usually stored in a ".cgx" file in the home directory to link them to the menu during startup. Regard also the command "subm" which uses submenus below 'user' for the same task.

9.61 merg

```
'merg' 'n'|'e'|'p'|'l'|'c'|'s' <set> <gtol> 'nolock'
```

This keyword is used to merge close points and nodes or equally defined entities. The following entities are known: Nodes n, Elements e, Points p, Lines l, Combined Lines c, Surfaces s. For example, to merge points included in the set point-set type

merg p point-set

Only entities included in the set are considered. The value gtol determines the maximum distance between merged nodes and points. The parameter nolock will force merging even if the dependent entities, like lines, are degenerated afterwards. For example, a degenerate line will have two equal points.

9.62 mesh

```
'mesh' <set> ['fast'] ['tet' <size>|'block'|->
              'lonly'|'nolength'|'noangle'|'length'|'angle']
```

This keyword is used to start the meshing of the model. before using the mesh command, the element types must be defined with the elty command. Existing elements will not be deleted. Therefore, its possible to start cgx in the viewing mode (-v) with a mesh alone, and then create bodies and fill them with additional elements. To delete a mesh use the command del mesh. See the command "send" to describe areas for boundary-conditions.

In case a blocked grid for cfd-calculations should be generated, use the additional parameter "block":

mesh setname block

see also the section "Remarks Concerning Duns and Isaac".

The mesh optimizer for structured elements is controlled with the additional

parameters "nolength" and "noangle". These parameters switch off length and angle optimizations of elements. The parameter "fast" suppresses the Nurbs-based meshing and uses the fast coons-algorithm for surface meshing.

Usually all elements are generated in a structured way. That means that no holes in a surface or volume are permitted. The only exception are the element types "tr3u" and "tr6u". This types are generated in an unstructured way using the mesher from [15]. In this case, holes are permitted and a surface coming from a cad system should be meshable. These surfaces MUST reference a given NURBS-surface or a shape to be meshable (if they are plane, a shape will be genrated atomatically). This is usually the case if the data is derived from a cad-system with the interface-program (vda2fbd). If NETGEN [4] is installed and if the elements form a closed volume they can be used to generate a tet mesh:

mesh set-with-trias tet

or

mesh set-with-trias tet <element-target-size>

This is a second method to generate tets. The other one is to use "elty" to asign tet-elements to bodies. Remark: The mesher is able to use multi threading. The number of cores are defined in cgx.h (NTHREADS) and can be changed with "asgn" during run time. The actual number can be listed with "prnt info"

9.63 mids

```
'mids' <set> ['force'|'lin'|'gen'|'rem']
```

This keyword is used correct the midside node-position of higher order elements stored in a set. It is performed automatically if a new mesh is created or if nodes are projected to target surfaces. The correction will use a circular track defined by the corner-nodes and the midside node. With the parameter "lin," the corrected position is halfway between the corner-nodes for all inner nodes. Except the midside nodes on the surface are not linearized but use the circular track. The user might overrule this restriction with the "force" option. The "gen" option will generate midside nodes for linear elements like he8 or te4. The "rem" option will remove midside nodes from the element formulation but the nodes are not deleted. The nodes are stored in a new set called "-delete" and the user might "zap" this set.

9.64 min

```
'min' <value> ['e'|'f'|'i'|'l'] ['l'|'u']
```

This keyword is used to define the lower value in the scale in the menu area (see figure 2). The number representation can be changed between exp, float, int or

log10 for which the scale is restricted to positive values. For example

min 0 i

will set the lower value to 0 and the representation to integer. A third parameter "l" (lock) or "u" (unlock) can be provided which locks the scale to certain max or min values. The selection of a different dataset will not change the scale.

9.65 minc

```
'minc' <value> ['e'|'f'|'i'|'l'] ['l'|'u']
```

The functionality of "minr" but the color of the first box is 'n' (neutral) unless the user had changed this color by using the asgn command.

9.66 minr

```
'minr' <value> ['e'|'f'|'i'|'l'] ['l'|'u']
```

The functionality of "min" but the specified value defines the lowest value of the second box.

9.67 minus

```
'minus' 'n'|'e'|'p'|'l'|'s'|'b'|'S'|'L'|'sh' <set>
```

This keyword is used to remove entities of a set from the screen (see also plus). The following entities are known:

Nodes n, Elements e, Points p, Lines l, Surfaces s, Bodies b, Nurb Surfaces S, Nurb Lines L and Shapes sh

Only the set which was used to display the entities can be used to remove them.

9.68 mm

```
'mm' <value> ['f'|'i'|'e'] ['l'|'u']
```

This keyword combines the functionality of the commands max and min in one command. The minimum value is set to -max.

9.69 move

```
'move' <set> ['scal' <fx> <fy> <fz> <pnt>]|
             ['tra' <dx> <dy> <dz>]|
             ['rot' [<p1> [<p2>]|['x'|'y'|'z']] |
                    ['x'|'y'|'z'] <alfa> |
                    [<alfa1> <ax1> <alfa2> <ax2>]]|
             ['rad' [<p1> [<p2>]|['x'|'y'|'z']] |['x'|'y'|'z'] |
                  'p'<pnt> <dr>] | [<dr1> <ax1> <dr2> <ax2>]
             ['nor' <dr>]|
             ['equ' <trgt-set> [<tol>]]|
             ['mir' <P1> <P2>]
```

This keyword is used to move nodes or points which are stored in a set. Related entities will be moved as well (Warning: results are not affected by this command, they are unchanged). For example to move a line it is necessary to include their points in a set (see comp). Several transformations are available:

For example scal will scale the entities of the set, the scaling factors fx, fy, fz can be chosen independently and a reference point can be used,

move part scal 2
move part scal 1 1 2

move part scal 2 P0
move part scal 1 1 2 P0

tra will move it away by the vector dx, dy, dz,

move all tra 10 20 30

rot will move it around the axis defined by the points p1 and p2 (or the axis x,y,z) by alfa degrees,

move all rot p0 px 20.

rad will move it radially to the x-, y- or z-axis (or two points as above) or to a single point,

move cylinder rad x 20.
move sphere rad pP0 10.

The axis for the rad or rot commands can also be specified by one point and on main-axis (x—y—z) as shown in the following example:

move all rot P y 20.

The change in length or angle might be interpolated for the rad or rot cases:

move set rad x 1. 120. 2. 140.

The number 1 specifies the radial change around x at 120 length units along the x-axis and 2 is the change at 140 length units,

nor will move nodes away in the direction of averaged normal local vector. Associated element faces must exist. Eventually use the "mids" command to correct the midside node position of higher order elements,

move set nor 1.2 a

With parameter equ points or nodes will be moved to their nearest neighbour (nodes only to nodes, points only to points) in set "trgt-set" as long as the neighbour is not more than 0.01 units away:

move slave-set equ trgt-set 0.01

mir will mirror the set. The mirror-plane is placed normal to the direction running from P1 to P2 and placed at P2, or defined by a point and a main-axis (x—y—z) as shown in the following example:

move part mir P y

9.70 movi

```
'movi' [loops <nr>]|[delay <sec>]|[start]|[stop]|
       [frames ['auto']|[<nr> [<epilogFile>]]]|
       [make [<pic-nr> <pic-nr> [<prolog.gif>]]]|
       [clean]
```

This keyword is used to start or stop the recording of a movie. After "start" all frames will be stored in single gif files until the "stop" command is issued. Use the option "make" to assemble the movie from the individual files. The range consists of the nr of the first and last picture to be used. An existing movie will be copied in front of a range of frames if its name is given. With the option "delay" a time-delay (in seconds) between frames can be specified. With the option "loops" a certain nr of loops can be chosen before the animation stops. Without giving a certain nr the default is chosen which is infinite loops. With the option "clean," all single gif-files will be erased.

Below is an example command sequence. Do not use this sequence in a file since the start and stop commands will be executed without delay (see option 'frames' for use in a command file). Instead of using the default value of loops here one loop is defined:

```
movi delay 0.01
movi loops 1
movi start
(let the program run until all frames are recorded)
movi stop
movi make
(or if a certain movie should be extended by the first 500 frames:)
movi make 1 500 prolog.gif
movi clean
```

When using the "frames" option the recording starts and a given nr of frames will be recorded before the recording stops atomatically. In cases were an animation of a mode shape or a sequence of datasets should be recorded it might be useful to use the argument 'auto' instead of a specific nr of frames. With the 'auto' functionality the program determines how much frames are needed to cover one period of frames and this period is then recorded. The 'make' and 'clean' functionality is included in the 'auto' mode. The 'auto' mode requires that the animation or the sequence is defined and started with the next command line (see "ds"):

```
anim real
movi frames auto
ds 3 eh 7
```

There is a second method available when successive commands after the recording of a given number of frames are needed:

movi frames 90 epilogCommandFile.fbl

This command must be the last command in an eventual command file. After 90 frames the given file 'epilogCommandFile.fbl' will be executed (the records are interpreted as cgx commands).

Further remarks in "How to change the format of the movie file". See also the menu options "Start Recording Gif-Movie".

9.71 msg

```
'msg' 'on|off'
```

This keyword is used to enable or disable full printout during runtime. This is useful for debugging purposes. The default is "off".

9.72 mshp

```
'mshp' <name> 'l'|'s'|'b' ->
   <element-type-nr> <element-attr-nr> ->
   <lockFlag> <density>|<size>
```

This keyword is used to set the mesh parameters for individual surfaces and bodies:

mshp A001 s 8 -1 1 2.074

sets the lement type to 8 (please see "Element Types" for a key to the element numbers), the attribute to -1 (tr6u) and the mesh density to 2.074 (mesh refinement). The '1' after the '-1' indicate a locked state and that all this properties can not be changed with an "elty" command before unlocked. For sets of surfaces or bodies the elty command must be used. The attributes are integer values with the following meaning:

- -1: unstructured mesh tr3u (-2 for mesh with libGLu tr3g)
- 0: default
- 1: reduced integration be2r be3r he8r he20r
- 2: incompatible modes he8i
- 3: DASHPOTA be2d
- 4: plane strain (CPE) tr3e tr6e qu4e qu8e
- 5: plane stress (CPS) tr3s
- 6: axisymmetric (CAX) tr3c
- 7: fluid he8f
- 8: tet10m
- 14: reduced integration, plane strain (CPE)
- 15: reduced integration, plane stress (CPS)
- 16: reduced integration, axisymmetric (CAX)

9.73 neigh

```
'neigh' <set> <tol> ['abq'|'ans'|'nas'] ->
         ['con' ['tie']|[[<stiffness>] [<mue>]]]|
         ['equ' [<dofs('t'|'1-6')..> 'c'|'u'..]|
         ['tie' ['yes']]|
         ['nsc' ['tie']|[[<stiffness>] [<mue>]]]
```

This keyword is used to find neighboring element faces (and nodes) which can be used in a contact formulation. So far *TIE, *CONTACT PAIR, or *EQUATION formulations for abaqus and ccx are availabe but only equations for ansys and nastran. It will search for disjunct meshes and generates sets storing the faces of this meshes with setnames starting with '+CF'. The neighboring element faces

are stored in additional sets which reference this meshes. The name of such a set consist of three parts. The first part of the name is just one character indicating if it is to be used at the dependent 'D' or independent 'I' side. The second part references the set containing the dependent faces, the third references the neighbor (the leading '+' of the basic sets are neglected). For example:

```
DCF2_CF4
```

includes the faces and nodes of '+CF2' which are close to '+CF4' were '+CF2' should be the dependent side. All sets for which no partner could be found are stored in set '+UNCON'.

The cgx writes equations connecting both sets when using the optional parameter 'equ':

neigh all 0.1 abq equ

But the recommended method is using 'tie', 'con' or 'nsc' which write the ccx command *TIE;

neigh all 0.1 abq tie

or *CONTACT:

neigh all 0.1 abq con 1e6 0.2

In this example the value 1e6 is used as normal contact stiffness (1/100 of that will be used as tangential stiffnes) and 0.2 as friction coefficient. The parameter 'con' defines surface to surface contact and 'nsc' node to surface contact. If the user does not provide values for stiffness and mue the cgx does not write *SURFACE INTERACTION, *SURFACE BEHAVIOR and *FRICTION commands. If instead of a stiffness 'tie' is given then a tied contact is formulated:

neigh all 0.1 abq con tie

Since ccx regards no gap criterion to exclude faces which are not in contact the surface sets must be defined more restricted for the 'con tie' option. Because of that some close faces might not be included in the contact sets. REMARK: Option TIED with node to surface contact does not work so far, so an alternative method is recomended:

Instead of PRESSURE-OVERCLOSURE=TIED the cgx writes PRESSURE-OVERCLOSURE=LINEAR with a normal and tangential stiffness of 1e7. To prevent a sliding contact mue is set to 1e30. The user should run his calculation in the following way:

```
** generate springs
*STEP
*STATIC
```

```
*BOUNDARY
Nall,1,3
*END STEP
** use the springs w/o updating them
** either:
***STEP,NLGEOM,perturbation
** or:
*STEP,perturbation
** either:
*STATIC
** or:
*FREQUENCY
*BOUNDARY,OP=NEW
```

).

The *TIE commands use the parameter ADJUST=NO for robustness (might be overruled when using as a last parameter 'yes'):

neigh all 0.1 tie yes

The user should check his calculation for unrealistic high stresses at the junctions and eventually change the formulation on such locations to ADJUST=YES. If the calculation does not run anymore he might modify the mesh locally or use the 'equ' option for that location.

The necessary ccx contact commands and the set definitions are written when using the parameter 'tie', 'nsc' or 'con'. The single files storing the sets might be bundeled into one file with:

cat CF* >| all.inp

This file and the file with the *TIE or *CONTACT formulations are then referenced in the ccx input file with *INCLUDE commands:

```
*INCLUDE, input=all.inp
*INCLUDE, input=neigh.con
```

When using the 'equ' option just one file has to be generated and used in ccx:

```
*INCLUDE, input=all.inp
```

It should be mentioned that the option 'equ' uses actually the same functionality as the command "send" with parameter 'areampc'. In consequence all the functionality of that command is usable. A closer description of the available options can be found in that section. For example the last parameter 'u' prevents the adaption of the position of the dependent nodes.

The command writes the setnames of the dependent and independent face pairs to the "stack".

9.74 node

```
'node' <nr|!> [<x> <y> <z> ['0'|'1']]|
              ['v' <value> [<value> .. ]]|
              ['vs' <value> ]
```

This keyword is used to define or redefine a node. For example

node 23580 10. 0. 1.

defines a node with the number 23580 at the position x=10, y=0, z=1. With a trailing '0' the update of related entities is skipped:

node 23580 10. 0. 1. 0

This is useful to speed up the execution when a long list of nodes will be defined. Just the last node needs the update which is executed with a trailing '1'. But that is the default anyway and can be omitted.

If the an automatically generated name is desired, then type "!" instead of a name.

The values of the selected (current) dataset of a given node can be manipulated by using the option 'v':

node 23580 v 1.1 2.1 3.3

redefine the first three entities to the above values of the selected dataset. The maximum number of entities is determined by the definition of the dataset. See also "ds" on how to select or generate a new dataset or on how to manipulate entities of the selected dataset.

The value of the selected (current) entity and dataset of a given node can be manipulated by using the option 'vs':

node 23580 vs 1.1

9.75 norm

```
'norm' <set>
```

This keyword is used to evaluate the normal direction of nodes stored in the given set. Of course the node must be referenced by faces. The command writes the normal direction of nodes to the konsole. The user might store this values on the stack for further use (see also stack and valu).

9.76 nurl

```
'nurl' <name(char9)>|'!' ['DEFINE' ['COMPACT'] ->
```

```
                <pstart> <pend> <deg> <npnt> <nknt> <div>]|
                ['CONTROL' <index> [<pnt>|<x y z>] <weight>]|
                ['KNOT' <index> <value>]|
                ['END']
```

This keyword is used to define a nurbs line. So far, this command is only used to read a nurbs-line definition. Nurbs lines are converted automatically into a spline. Nurbs lines can be displayed but not saved. There are two possible ways of definition. Either by using predefined point-names or by specifying the coordinates explicitly. When the coordinates are defined, the parameter "COMPACT" must be used as shown above. When the point names are used, then "COMPACT" must be omitted. CAD-interfaces might use this functionality.

9.77 nurs

```
'nurs' [<name(char9)>|'!' ['DEFINE' ['COMPACT'] ->
           <u.deg> <v.deg> <u.npnt> <v.npnt> <u.nknt> <v.nknt>]|
           ['CONTROL' <u.index> <v.index> [<pnt>|<x y z>] ->
           <weight>]|
           ['KNOT' <U>|<V> <index> <value>]|
           ['END']] |
       [ <!> <setname(containing surfaces)>]
```

This keyword is used to define a nurbs shape. Surfaces might use nurbs to define the interior geometry. There are two possible ways of definition. The first using predefined point names and the second by specifying the coordinates explicitly. When the coordinates are defined, then the parameter "COMPACT" must be used as shown above but when point names are used, then "COMPACT" must be omitted. CAD-interfaces might use this functionality.

REMARK: The knot-vector has to have a multiplicity of "degree+1".

There is also a small nurbs-building capability in cgx. It is possible to use existing surfaces (with 4 edges) which do not already reference a given nurbs or shape. The new nurbs will follow the Coons-algorithm but can be modified by moving the control points. NOTE: The number of control points is controlled by the divisions of the lines defining the surface edges. The surfaces must be stored in a set. For example,

nurs ! surfaceSet

will define nurbs for all surfaces stored in the set surfaceSet. This nurbs can be used to define the interiour of other surfaces. This is necessary if "tr3u" elements (unstructured triangles) should be used and if the surface is not related to a given shape. Note: "qsur" offers another option to create nurbs related surfaces by associating existing surfaces to an overlapping existing NURBS. The interiour of the surface is then defined by the NURBS.

Remark: Internally nurbs are always linked by a shape to a surface definition. Such a shape will be automatically generated when a nurbs is finished by the "END" parameter using the same name as the nurbs. This shapes will not be written to a file but using the prnt command will list them:

shpe N001 NURS N001

9.78 ori

```
'ori' <set>
```

This keyword is used to trigger the orientation of the entities. This is done automatically and it should never be necessary to use it manually.

9.79 plot

```
'plot' ['n'|'e'|'f'|'p'|'l'|'s'|'b'|'S'|'L'|'sh'|'si']&->
       ['a'|'b'|'c'|'d'|'n'|'p'|'q'|'t'|'v'] ->
       [<set>|<*chars*>] ['b'|'g'|'k'|'m'|'n'|'r'|'t'|'w'|'y'] [<width>->
             |<transparency>]
```

This keyword is used to display the entities of a set. Entities already visible will be erased. The following types of entities are known:

Nodes n, Elements e, Faces f, Points p, Lines l, Surfaces s, Bodies b, Nurbs Surfaces S, Nurbs Lines L, Shapes sh and the shaded (illuminated) surfaces si

The entities can be displayed in the following colors:

White w, Black k, Red r, Green g, Blue b, Yellow y, Magenta m, Neutral 'n' (metallic grey) and turquois t

To display the entities with attributes, use the type in combination with an attribute (second letter). If no set is specified, the entities of the given type of all sets are shown:

plot e

shows the elements of all sets in the available colors (except set 'all'). The command

plot e *

does the same. In general wildcards(*) are supported:

plot e E*

will display all elements of all sets staring with an 'E'.

To get a list of the visible sets and the used colors use "prnt ve".

Other examples are:

plot la all

will display all lines with their names. The attribute d works only for lines,

plot ld all

will display all lines with their division and bias (see bia). The division is given by the numbers (1-99) following the # sign and the bias by the leading digits. If the bias uses more than one digit in front of the division, the number has to be divided by a factor of ten to get the bias (101#30 means a bias of 10.1 and a div of 30).

plot ln all

shows potential node locations. The attribute p works only for lines. In this case the lines with its end-points are drawn:

plot lp all

This is useful to detect the begin and end of all lines. If end-points are deleted, the line is also deleted. Therefore special care with end-points is necessary. The key c combines the line parameters d, p, n:

plot lc all

The lines are drawn with their end-points, potential node positions and divisions. Shaded surfaces

plot si all

can only be displayed if the interiour was previously calculated, which is done with the command "rep" or "mesh". The attribute t applies only to nodes and will display only the ones which have attached texts:

plot nt all

will display only the nodes which have attached texts out of the set 'all'. They are created with "qadd", "qenq" or "qtxt". The attribute "width" determines the number of pixels used for the thickness of the entity (points, nodes, lines):

plot l all 4

will display all lines with a width of 4 pixels. This works also for 2D faces and beams. The attribute n works for nodes only:

plot nn set1

will display the nodes in set set1 with their numerical values. The attribute v works for nodes, faces ane elements. This attribute is used to display results with colors representing their values:

plot nv set1

plot fv set1

plot ev set1

Actually this is what happens automatically if the user selects an "Entity" from "Datasets" in the "menu". The faces can be displayed in a transparent manner with the attribute b:

plot fb set1 t 33

will display the faces in turquois color with a transparency of 33%.

plot fvb set1 33

will display the faces with their colored values with a transparency of 33%. A default transparency is used if a number is not given.

The attribute q works only for elements. With this attribute, only elements which do not pass the element-quality check are displayed:

plot eq all

The threshold for the element-quality is defined with "eqal".

To plot additional entities, see plus.

9.80 plus

```
'plus' ['n'|'e'|'f'|'p'|'l'|'s'|'b'|'S'|'L'|'sh'|'si']& ->
       ['a'|'b'|'d'|'n'|'p'|'q'|'t'|'v'] ->
       [<set>] ['b'|'g'|'k'|'m'|'n'|'r'|'t'|'w'|'y'] [<width>->
             |<transparency>]
```

This keyword is used to display the entities of an additional set after a plot

command was used (see also minus). Further details are explained in section plot.

9.81 pnt

```
'pnt' <name(char<9)>|'!' [<x> <y> <z>]|
                         [<line> <ratio> <times>]|
                         [<P1> <P2> <ratio> <times>]|
                         [<setname(containing nodes)>]
```

This keyword is used to define or redefine a point. There are four possibilities to define a point. To define a point just with coordinates:

pnt p1 11 1.2 34

or,

pnt ! 11 1.2 34

where the name is chosen automatically. It is also possible to create points on a line or in the direction from P1 to P2 by defining a spacing (ratio) and number-of-points:

pnt ! L1 0.25 3

or

pnt ! P1 P2 0.25 3

will create 3 new points at the positions 0.25, 0.5 and 0.75 times the length of the line or the distance from P1 to P2, and it is also possible to create points on the positions of existing nodes. The command

pnt ! set

will create new points on the positions of the nodes included in the specified set. Usually when points are defined interactive the command qpnt is used.

9.82 prnt

```
'prnt' ['se'|'sq' <RETURN|set|*chars*> ['range']]|
       ['n'|'e' <set|*chars*> 'range']|
       ['n'|'e'|'f'|'p'|'l'|'s'|'b'|'L'|'S'|'v' <entity>]|
       ['col']|
       ['amp' <RETURN|amplitude|*chars*>]|
       ['mat' <RETURN|material|*chars*>]|
```

```
['par' <RETURN|parameter>]|['usr']|
['eq' <set>]|
['st' ['si']]|
['in']|['ve']|
['capt']|['ulin']
```

This keyword is used to print entity-definitions. The following entities are known:

Nodes n, Elements e, Faces f, Points p, Lines l, Surfaces s, Bodies b, Nurb Lines L, Nurb Surfaces S, Values v, Sets se and Sequences sq

To see all known sets, type:

prnt se

Or type

prnt sq

to see all known sequential sets (sequences: Sets which maintain the history of entity selection). Or type

prnt ve

to list all sets used to show the visible entities.

Wildcards (*) can be used to search for setnames of a certain expression. In this case all sets matching the expression will be listed:

prnt se N*

lists all sets starting with 'N'. To see the contents of a specific set, type

prnt se setName

Please see "mshp" for details about etyp, attr and "elty" about lock. In case the "stack" was activated the value and the coordinates of stored nodes are written to the stack. The content of the stack can be listed with:

prnt st

The index and the value are written. The value with the highest index is addressed next. The number of stack entries (the size) can be enquired (and written to the stack) with:

prnt st si

A model summary (info) is listed with:

prnt in

which will list the number of mesh- and geometric entities. In addition it lists the hidden entity 'edge' which are the free element edges. This number has to be zero in case of a closed surface triangulation which is required for tet-meshing.

To print the definition of a line, type:

prnt l lineName

An eventually assigned alias name for a given entity can be enquired with a leading question mark:

prnt l ?lineName

For elements and nodes is an additional parameter "range" availabe. In this case the range of node- or element-numbers will be displayed (max and min nr) and holes in the numbering are detected:

prnt n setname range

If an ccx- or abaqus-input-file was read then it is also possible to print the amplitudes (*AMPLITUDE in ccx) or the material-propperties (*MATERIAL in ccx), wildcards (*) can be used:

prnt amp amplitude-name

prnt mat material-name

If an ccx result file was read then the user headers (meta data)

prnt usr

and parameters of each dataset can be listed and written to the stack with either

prnt par

to list all parameters of the active dataset or a single one like for example

prnt par STEP

can be listed and written to the stack (see stack). The available colors are listed with

prnt col

See “col” on how to define or redefine colors.

The element quality is checked by using

prnt eq setName

The failed elements are stored in set “-NJBY”. See “eqal” on how to set the criterions.

With the parameter ’capt’ the program writes the captation (see command capt) to the terminal and to the stack:

prnt capt

The same is true for the parameter ’ulin’ (see command ulin):

prnt ulin

Both are text lines below the graphic 3D window.

Several other but not all parameters of this command writes to the ”stack”.

9.83 proj

```
’proj’ <set> <target-set>|<shpe> ->
           [’tra’ <dx> <dy> <dz> <offset> [<tol>]]|
           [’rot’ <p1> <p2> <offset> [<tol>]]|
           [’rot’ ’x’|’y’|’z’ <offset> [<tol>]]|
           [’rad’ <p1> <p2> <offset> [<tol>]]|
           [’rad’ ’x’|’y’|’z’ <offset> [<tol>]]|
           [’nor’ <offset> [<tol>] ]
```

This keyword is used to project points (with all related geometry) or nodes onto a set containing nurbs, shapes, surfaces or element-faces. Alternatively the name of a shape can be specified as the target.

Several transformations are available. For example tra will move points in the direction of the vector dx, dy, dz onto elements or surfaces included in set2. Alternatively an offset could be specified as well,

proj set1 set2 tra 0. 0.5 0.7

rot will move points around the axis defined by the points p1 and p2 or around the x,y,z axis onto elements or surfaces included in set2,

proj set1 set2 rot p0 px

rad will move points radial to the axis defined by the points p1 and p2 or

radial to the x-, y- or z-axis onto elements or surfaces included in set2. Alternatively a set of lines could be used instead of surfaces as the target-set. Then the geometry will be moved onto an imaginary rotational surface defined by these lines,

proj set1 set2 rad x

nor will move points in a direction normal to the target surface onto surfaces included in set2. An offset might be specified:

proj set1 set2 nor 0.7

If a shape was given instead of a target-set then an offset can not be used:

proj set1 ShapeName nor

If a point does not hit any surface from the target-set, then it will not be moved. A tolerance can be specified for the projections. No projection takes place if the target surface is farther away as this value defines:

proj set1 set2 nor 0.7 10.

Check for set '-NOPRJ'. If it exists it will store the failed nodes or points.

9.84 qadd

```
'qadd' <set|seq> ['t'<value>] RETURN ->
       <'w'|'a'|'i'|'r'|'n'|'e'|'f'|'p'|'l'| ->
        's'|'b'|'S'|'L'|'h'|'m'|'q'|'s'|'t'|'u'>
```

This keyword is used to add entities to a set. See also seta. But a set will not keep the sequence in which the entities were selected. Use seqa or the command "qseq" if the order of the selected entities has to be kept.

After an entitiy was selected you get certain informations about the entity. If the node which belongs to the maximum or minimum value in a certain area has to be stored in a set you might use the same key-strokes as described for the command "qenq".

To catch more than one entity with one stroke, type "a" (all) at first. Then create a rectangular picking area by pressing two times the "r" key. Both strokes define opposite corners of the selection-rectangle. To catch only the entity which is closest to the user type "i" before.

Then move the mouse pointer over the entity(s) and press one of the following keys, for Nodes n, Elements e, Faces f, Points p, Lines l, Surfaces s, Bodies b, Nurb Surfaces S, node attached texts t and for Nurb Lines L.

If faces f of a certain area have to be selected, the user might specify a tolerance-value which restricts the deviation of the normal vectors of faces from the selected face. As long as the deviation is below the specified value (in degrees) all adjacent faces will be selected in a loop:

qadd areaset t25

Press "q" to quit the command.

It is also possible to measure distances between two pixels on the screen. Just press the key "w" on the positions of the two pixels. The distance is calculated in the scale of the displayed geometry.

9.85 qali

```
'qali' RETURN 'w'|'p'|'n'|'q'
```

This keyword is used to align a plane defined by three points or nodes with the screen (working plane). This is useful if a point or a node should be moved manually along a defined plane (see qpnt). To define the plane move the mouse pointer over the first entity and press "n" if its a node or a "p" if its a point. Then define the next two entities in the same way. Press "q" to quit the command.

It is also possible to measure distances between two pixels on the screen. Just press the key "w" on the positions of the two pixels. The distance is calculated in the scale of the displayed geometry.

9.86 qbia

```
'qbia' RETURN 'w'|'a'|'i'|'c'|'1'-'9'|' 10'-' 99'|'q'
```

This keyword is used to change the bias of a line. The bias defines a coarsening or refinement of the mesh along a line. The number defines the ratio of the length of the first element to the length of the last element at a given line. It works by pressing a number between 1 and 9 when the mouse pointer is at the position of a line. To define numbers between 10 and 99 press the space bar when the mouse pointer is at the position of a line and then the number (two times the space bar for 100 to 999). To select more than one line with one stroke, type "a" before and create a rectangular picking area by pressing two times the "r" key. Both strokes define opposite corners of the rectangle. To select only one line type "i" before. Press "c" to change the direction of the bias. Press "q" to quit the command (see alse bia). Consider to split a line if you need a bias in both direction (see qspl).

9.87 qbod

```
'qbod' <name>(optional) RETURN 'w'|'b'|'a'|'i'|'r'|'s'|'g'|
                               'q'|'u'
```

This keyword is used to create a body (see also gbod and body). The user might specify a name in the command-line or by picking an existing body with the key "b". Otherwise the program chooses an unused name. It is possible to create the body out of five to seven surfaces which are needed to define a body or just of two opposite surfaces, but then these two surfaces must be connected on their corner points by lines. To be more precise only single lines or existing combined lines (lcmb) will be detected. If a combined line would be necessary but does not exist then the user should define a surface using this lines which will create the necessary combined-line. Other missed surfaces will be created automatically. To catch more than one surface with one stroke, type "a" before and create a rectangular picking area by pressing two times the "r" key. Both strokes define opposite corners of the rectangle. Type "s" to select surfaces. To catch surfaces individually type "i" before (its also the default). After selecting exactly six or two opposite surfaces press "g" to generate the body. Press "q" to quit the command or "u" to undo the last action.

It is also possible to measure distances between two pixels on the screen. Just press the key "w" on the positions of the two pixels. The distance is calculated in the scale of the displayed geometry.

9.88 qcnt

```
'qcnt' RETURN 'w'|'n'|'p'
```

This keyword is used to define a new center-point or -node by pressing "n" or "p" when the mouse pointer is at the position of a node or a point.

It is also possible to measure distances between two pixels on the screen. Just press the key "w" on the positions of the two pixels. The distance is calculated in the scale of the displayed geometry.

9.89 qcut

```
'qcut' RETURN 'w'|'q'|'n'|'p'|'u'|'v'|'x'|'y'|'z'|'s'
```

This keyword is used to define a cutting plane trough elements to visualize internal results (see figure 5). New elements are generated and stored in the set "-qcut" for this purpose. The plane is defined either by picking three nodes (select with key "n") or points (select with key "p"), or by just one node plus either a successive "v"-keystroke in case a dataset-entity of a vector was already selected or just an "x"-,"y"- or "z"-keystroke without pressing 'n' before. The cutting plane is then determined normal to the vector (displacements, worstPS ..) or the given x,y or z direction. Be aware of the key "u" (undo) to return to the un-cutted structure. See also "cut" for the command-line function. The elements generated by successive qcut or cut add up until they are deleted.

To suppress or enable the immediate ploting of the cut press "s" (switch ploting on/off) before you create the cut. This is necessary if more than one cut should be shown together. So first press key "s" then create all the cuts except the last. Then "s" again and create the last cut.

It is also possible to measure distances between two pixels on the screen. Just press the key "w" on the positions of the two pixels. The distance is calculated in the scale of the displayed geometry.

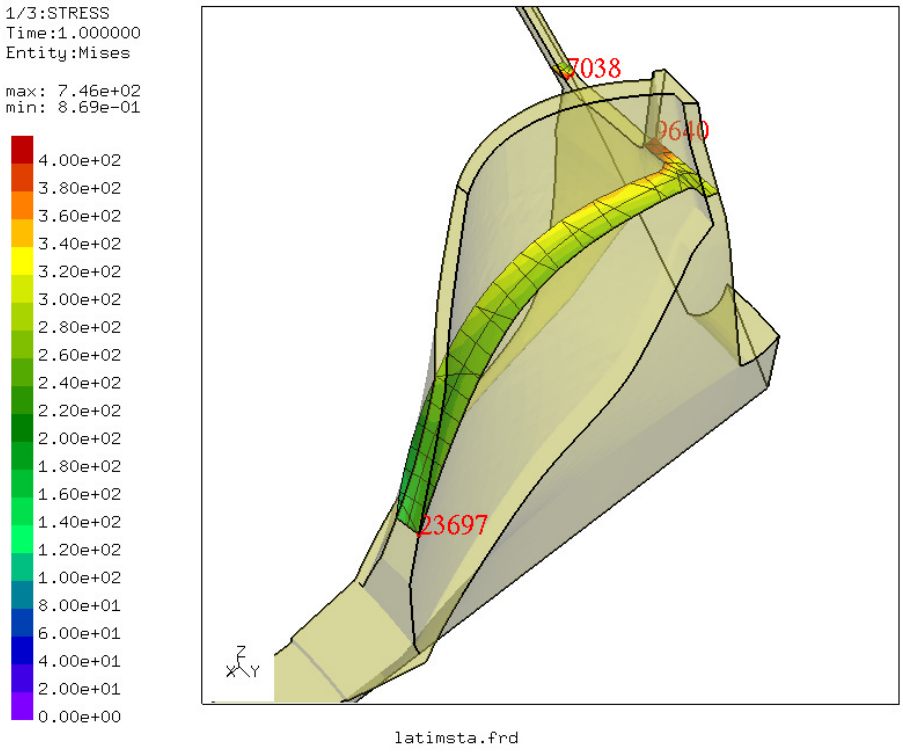


Figure 5: qcut: A section through a model defined by three nodes together with a transparent view of the outer skin

9.90 qdel

```
'qdel' RETURN 'w'|'a'|'i'|'r'|'p'|'l'|'s'|'b'|'S'|'L'|'h'|'q'
```

This keyword is used to delete entities (see also del). Higher entities (depending ones) will be deleted to. To delete more than one entity with one stroke, type "a" before and create a rectangular picking area by pressing two times the "r" key. Both strokes define opposite corners of the rectangle. To delete only one entity type "i" before. Press "q" to quit the command.

Surfaces will not be deleted when the user deletes lines which define complete holes in existing surfaces. But it is necessary that all lines of a certain hole are deleted at once. If only a partial hole is deleted, the surface is deleted as well. It is proposed to use zap to delete the affected lines.

It is also possible to measure distances between two pixels on the screen. Just press the key "w" on the positions of the two pixels. The distance is calculated in the scale of the displayed geometry.

9.91 qdis

```
'qdis' RETURN 'c'|'f'|'g'|'m'|'n'|'p'|'q'|'s'|'w'
```

This keyword is used to measure distances between nodes or points. Move the mouse pointer over one entity and press the following key, for a node 'n', for a point 'p' or for a center point 'c' (has to be the second selection). If the key 'c' was pressed then the distance between the two nodes or points are given in cylindrical coordinates. Here lcir is the length of the arc, da the angle, dr is r2-r1. If no center point was chosen then the distance and its xyz components are given in the Cartesian system. But in addition the cylindrical distances around the origin and around the xyz-axis are also given.

In addition the normal distance of a node or point to a plane defined by three nodes or points can be measured. The point or node in question is selected with key 'n' or 'p' and the nodes or points which define the plane are selected with 'm' (mesh nodes) or 'g' (geometry points). Instead of defining a plane the user may select a single surface with 's' or a single element face with 'f' (will be extended by adjacent faces). This selection has to take place before the node or point is selected.

Press "q" to quit the command.

- dist: global distance
- dx, dy,dz: distance in the three Cartesian directions
- da: global angle
- dax, day, daz: angle around x, y, z
- dr: global radius difference
- drx, dry, drz: radius difference around x, y, z

It is also possible to measure distances between two pixels on the screen. Just press the key "w" on the positions of the two pixels. The distance is calculated in the scale of the displayed geometry.

9.92 qdiv

```
'qdiv' RETURN 'w'|'a'|'i'|'1'-'9'|' 10'-' 99'|'q'
```

This keyword is used to change the division of a line by pressing a number between 1 and 9 when the mouse pointer is at the position of a line (see also div). To define numbers between 10 and 99 press the space bar when the mouse pointer is at the position of a line and then the number (two times the space bar

for 100 to 999). To select more than one line with one stroke, type "a" before and create a rectangular picking area by pressing two times the "r" key. Both strokes define opposite corners of the rectangle. To select only one line type "i" before. Press "q" to quit the command. General rules are described in "div".

It is also possible to measure distances between two pixels on the screen. Just press the key "w" on the positions of the two pixels. The distance is calculated in the scale of the displayed geometry.

9.93 qenq

```
'qenq' RETURN 'w'|'a'|'i'|'r'|'n'|'e'|'f'|'p'|'l'|'s'|->
              'b'|'S'|'L'|'h'|'m'|'u'|'v'|'t'|'q'
```

This keyword is used to gain information about entities. It is especially useful to get the values on particular nodes.

If the maximum or minimum value in a certain area has to be searched type first "m" to go in the max/min mode and create a rectangular picking area by pressing two times the "r" key. Both strokes define opposite corners of the selection-rectangle. Then the key "h" (high) to search the node with the maximum value. The minimum is searched with "l" (low). The search-result is then shown in the konsole and also attached to the node. With the "u" key the last search result can be deleted ("undo"). The 't'-key changes into the "qtxt"-mode. The "qtxt" functionality is now available which allows to manipulate the node-attached-string and its position on the screen. Use "qadd" instead of "qenq" if you need to save the node in a set for further use.

To catch more than one entity with one stroke, type "a" (all) before and create a rectangular picking area by pressing two times the "r" key. Both strokes define opposite corners of the rectangle. To catch only the entity which is closest to the user type "i" before.

Then move the mouse pointer over the entity(s) and press one of the following keys, for Nodes n, Elements e, Faces f, Points p, Lines l, Surfaces s, Bodies b, Nurb Surfaces S and for Nurb Lines L.

The position of nodes or points are given in Cartesian and cylindrical coordinates (see figure 6, axyz are the 3 angles around x,y and z, rxyz are the 3 radii around x, y and z). In a second row the sets to which the picked entity belongs are listed.

Press "q" to quit the command.

It is also possible to measure distances between two pixels on the screen. Just press the key "w" on the positions of the two pixels. The distance is calculated in the scale of the displayed geometry.

9.94 qfil

```
'qfil' <radius> RETURN 'w'|'l'|'q'
```

The command creates a fillet line (a radius) between two lines who intersect. After the command-name the value for the fillet-radius has to be specified. Then

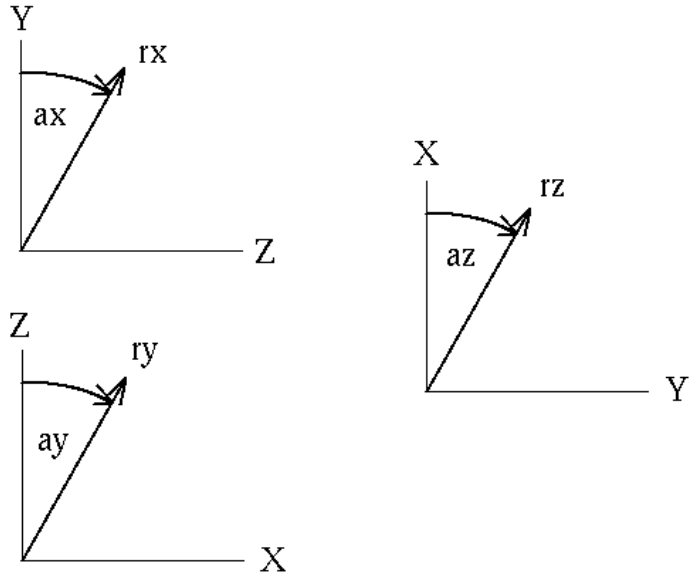


Figure 6: qenq: Definition of the cylindrical system

for each pair of selected lines a fillet of this value will be created. Select lines with the "l"-key. Press "q" to quit the command. Currently it works only for straight lines. A curved line can be split and the part which should be used for the fillet can be transformed to a straight form with the "qlin" command ("s" and "x" key-strokes). Figure 7 shows on the right side the initial situation and on the left side the created fillet. The command has also shifted point P002 to the left. Always the end-point of the first selected line closest to the intersection is moved. The second line gets a new end-point.

9.95 qflp

```
'qflp' RETURN 'w'|'e'|'s'|'a'|'i'|'q'
```

This keyword is used to invert the outer- and inner side (the orientation) of shell-elements and surfaces. The orientation of shell-elements or surfaces can be seen by the interior color. The outer face reflects light but the face is dark-grey if it is the back-side. If "Toggle Illuminate Backface" was selected before then both sides are illuminated. To flip the orientation of a surface and all related shell-elements select the illuminated surface (see "rep" how to do that) with the "s" key. To see the effect on the elements immediatelly they must have been displayed with the "plus" command as well. If only elements are in the

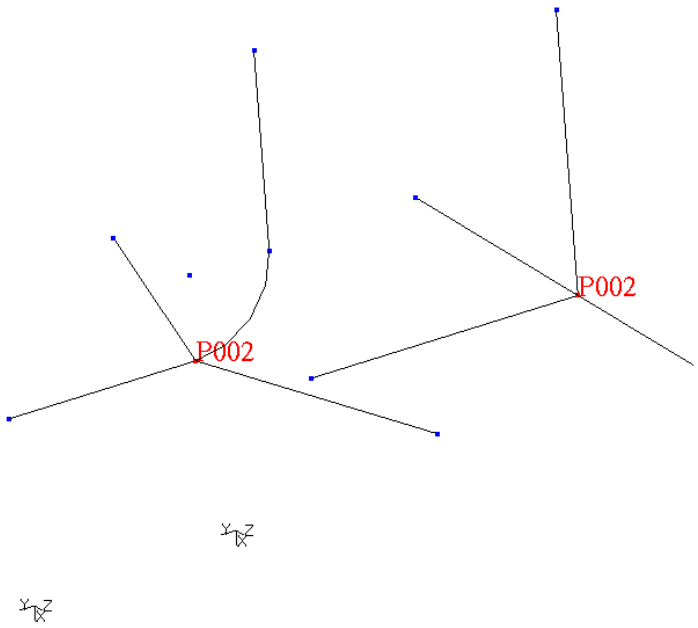


Figure 7: qfil: Based on two intersecting lines a fillet is created

database flip them with the "e" key. The "a" key has a different meaning than usually. If pressed before a selection then a so called auto mode is activated. In this case all related surfaces (and embedded elements) are oriented in the same way as the selected one. It works only in situations were only two surfaces share a common edge! This is the case for volumes without inner surfaces or a 2D model. In case of a volume all surface normals will point either inwards or outwards, depending on the orientation of the selected surface. Press "q" to quit the command.

It is also possible to measure distances between two pixels on the screen. Just press the key "w" on the positions of the two pixels. The distance is calculated in the scale of the displayed geometry.

9.96 qint

```
'qint' RETURN 'w'|'l'|'q'
```

This keyword is used to extend two lines to their intersecting location. The end-points of the two lines are moved to the intersecting location but they are

not merged. Use the "merg" command to merge duplicate points. Since only straight lines and splines are supported it is necessary to convert other line types into splines before this command can be used (see "qseq").

Select lines with the "l" key. Press "q" to quit the command.

9.97 qlin

```
'qlin' <name>(optional) RETURN 'w'|'b'|'c'|'e'|'g'|'l'|'m'|'p'|->
                               'q'|'s'|'t'|'u'|'x'
```

This keyword is used to create a sequence of lines (or just one, see also line). The user might specify a name in the command-line or by picking an existing line with the key "l". In this case an existing line might be redefined without destroying related geometries.

To start a sequence of lines move the mouse pointer over a point and press the key "b" (begin).

To define just a straight line go to the end point and press "g" (generate). This point is also the starting point for the next line. So no "b" is necessary for the next line.

If the line is a sequential line (spline) then the points of the sequence must be defined with the key "t" in the correct order, but the last point must be selected with the key "g".

If the line is an arc then the center point must be selected with the "c" key or one point on the line between the end-points must be selected with the "m" key, but the last point must be selected with the key "g".

In case you need to split a line during the process, you can do that by pressing the "s" key.

When you want to modify the length then pick an existing line with the key "l" and then the endpoint you want to move with the key "p". The displacement is requested and the point is moved in the direction of the line by the specified displacement.

The "x" key will transform the line into a straight line.

One quite useful function for cad-based surfaces is triggered by the "e"-key (exchange). This allows to modify an existing surface (gsur). With that command a line or a line sequence in a lcmb can be exchanged by the previous selected ("l" key) or generated line. For example one edge of an already existing surface uses a lcmb but the user wants to replace it by a spline (line) then the user defines or selects the new spline and then moves the mouse in an area were one of the old lines (the new one might be present as well) is located and presses the "e" key. The definition of the lcmb will be changed without destroying the surface-definition. Alternatively the existing lines can be combined with the "qseq" command into a single spline. This command will delete the original lines.

Press "q" to quit the command or "u" to delete the last created line.

It is also possible to measure distances between two pixels on the screen. Just press the key "w" on the positions of the two pixels. The distance is calculated

in the scale of the displayed geometry.

9.98 qmov

```
'qmov' <set> RETURN 'w'|'n'|'p'|'m'|'u'
```

This keyword is used to move a set of entities (see also move). To move a set move the mouse pointer to the desired reference node and press "n" or a reference point with "p". Then go to the desired position, either a second node, a point or a window location and press either "n" (node), "p" (point) or "m" (move). Press "q" to quit the command or "u" to undo the last action.

It is also possible to measure distances between two pixels on the screen. Just press the key "w" on the positions of the two pixels. The distance is calculated in the scale of the displayed geometry.

9.99 qmsh

```
'qmsh' RETURN 'i'|'a'|'e'|'f'|'v'|'n'|'l'|'c'|'s'|'m'|'d'|'h'|->
              't'|'x'|'q'
```

A very useful command to optimize surface meshes as a basis for tet-meshes. Line divisions can be changed by selecting one or more lines with the "l" key and then a number as with "qdiv". The mesh will be updated automatically.

Based on identified lines "l" or surfaces "s" meshes can be coarsened 't', refined 'h', deleted "d" and generated "m". Surfaces can be combined "c" to get rid of very small ones and splitted 'b'.

To change the mesh density along a line:

- Identify a line or several lines with key "l". The related surfaces are selected.
- The line division is changed by pressing the key with the desired number. This function of qmsh works like the "qdiv" command. The related surfaces are automatically re-meshed.
- The line bias is changed if first the key "b" is pressed and then the key with the desired number or the key "c" to inverse the direction. This function of qmsh works like the "qbia" command. The related surfaces are automatically re-meshed.

Example for a surface combination:

- Identify a line with key "l". The related surfaces are selected.
- Or identify two adjacent surfaces with key "s" or by selecting two single elements "e" one for each surface. They must use the same lines at the junction.

- To combine the two selected surfaces press key “c” (Attention: The surface with the longer perimeter must reference a shape (a nurbs)). In the moment an automatic adaption of the embedded shape or nurbs is not done. The shape of the bigger surface is just used for the combined surface. It might be necessary to use a mapped mesh for the new surface if the embedded shape is too small (use tr6 instead of tr6u, switching is done with key “x”).

A surface can be splitted by a crossing line:

- First create the desired splitting line, for example with “qlin”.
- Type “qmsh” and identify the line with key “l”.
- Identify the surface with key “s” or by selecting a single element “e”.
- Press key “b” to break (split) the surface in two. The previously selected line will be converted to a spline with as much control points as the line division. Then the line-points are projeted to the surface.

To change the mesh density in a selected surface:

- Select one or more surfaces with key “s” or by selecting lines “l” or by selecting a single element “e”.
- Press “h” (higher) to increase the mesh density or “t” (thinner) to decrease.
-

To change the element type from unstructured to structured or vice versa:

- Select one surfaces with key “x”. Be aware that only certain line division combinations will work for structured meshes. Initially the element type tr6 will be used if no element type was assigned to the selected surface.

This keyword can be used to manually define elements. The key “n” selects nodes and the key “f” or “v” generate a surface or volume element based on the node-selection.

To catch more than one entity with one stroke, type ”a” (all) at first. Then create a rectangular picking area by pressing two times the ”r” key. Both strokes define opposite corners of the selection-rectangle. To catch only the entity which is closest to the user type ”i” before.

9.100 qnor

```
'qnor' RETURN 'w'|'p'
```

A new line normal to a plane defined by three points and a length is created. It starts at the last point.

9.101 qpnt

```
'qpnt' <name>(optional)RETURN 'w'|'p'|'g'|'m'|'n'|'s'|'S'|'u'
```

This keyword is used to create or move points (see also pnt). The user might specify a name in the command-line if a certain name should be used. To create a point move the mouse pointer to the desired location and press the key "g" (generate) or over an existing node and press "n" (uses then the node-coordinates). After a point was selected with "p" it can be moved in different ways: Either in the screen-plane, for this go to the desired position and press "m" (move, see qali or the section Orientation how to rotate the model into a certain position). Or the point can be moved to the position of a second point, for this go to the second point which coordinates should be used and press "p" again. If the coordinates of a node should be used press "n" instead. Also a normal projection to a nurbs related surface (from cad-systems) is feasible by choosing the target-surface with the "s" key or a NURBS-surface with the "S" key either before or after the point was marked. Press "q" to quit the command or "u" to undo the last action.

If you picked the wrong point (the one which should be moved), just pick the same again and pick then the correct one.

It is also possible to measure distances between two pixels on the screen. Just press the key "w" on the positions of the two pixels. The distance is calculated in the scale of the displayed geometry.

9.102 qnod

```
'qnod' RETURN 'w'|'p'|'m'|'u'
```

This keyword is used to move nodes (see also node). To move a node move the mouse pointer to the desired node and press "p" (pick) then go to the desired position and press "m" (move). See qali or the section Orientation how to rotate the model into a certain position. Press "q" to quit the command or "u" to undo the last action.

It is also possible to measure distances between two pixels on the screen. Just press the key "w" on the positions of the two pixels. The distance is calculated in the scale of the displayed geometry.

9.103 qrem

```
'qrem' <set> RETURN 'w'|'a'|'i'|'r'|'n'|'e'|'f'|'p'|'l'|'s'|->
                    'b'|'q'
```

This keyword is used to erase entities from a set (see also setr). To remove entities move the mouse pointer over the entity and press the following keys, for Nodes n, Elements e, Faces f, Points p, Lines l, Surfaces s, Bodies b, Nurb Surfaces S and for Nurb Lines L. To catch more than one entity with one stroke type "a" before and create a rectangular picking area by pressing two times the

"r" key. Both strokes define opposite corners of the rectangle. To catch only one entity type "i" before. Press "q" to quit the command.

It is also possible to measure distances between two pixels on the screen. Just press the key "w" on the positions of the two pixels. The distance is calculated in the scale of the displayed geometry.

9.104 qseq

```
'qseq' [[<set>] RETURN 'p']|[ RETURN [<nr>|'l'|'g']]
```

This keyword is used to define a sequence of nodes or points or to convert one or more lines into a sequential line (spline). A sequence is nothing else than a set which keeps the selection-order of the entities:

qseq nodeset

will store all selected nodes (selected with the 'n' key) in the order in which they were selected. This set might be used for a subsequent "graph" command. Point sequences are used to define splines. Sequences can be shown with the "prnt sq" command. Node sequences can be written to a file and read again with the commands "send seqname fbd" and "read seqname.fbd".

To change one or more lines into a sequential line type

qseq

without a setname. Then either one existing line has to be selected with a numerical keystroke (or the space-bar followed by two numbers) or several lines with 'l' keystrokes. In both cases the selection has to be finished by a 'g' keystroke. New points will be created on the selected lines which in turn are used to redefine the selected lines as a sequence line. If just one line was selected then as much points are generated as defined with the nummerical keystroke. If more than one line was selected with the 'l' key then the combined line divisions minus one define the number of new points. This command will replace and delete the original lines.

9.105 qshp

```
'qshp' RETURN 'w'|'n'|'p'|'g'|'h'|'s'|'S'|'c'|'q'
```

This keyword is used to define a plane shape. A plane shape needs 3 points (or nodes, points at their location will be generated and used) for its definition. The points are selected with the 'p' (nodes 'n') key. After the points are selected it will be generated by a 'g' keystroke. The shape can be assigned to a surface by selecting first either a shape 'h' or a nurbs 'S' and then the surface with the 's' key. The shape will then define the interiour of this surface. The selected shape or nurbs stay selected until cleared with 'c'. See also "shpe" for the keyboard controlled definition of shapes.

It can also be used for projection purposes (see "proj") or splitting (see "split").

9.106 qspl

```
'qspl' RETURN 'w'|'s'|'q'
```

This keyword is used to split one or more lines at a certain position. A point is created at the splitting position, the original line is deleted and two new lines will appear instead. All lines running through the selected location are splitted at once and the newly created splitting points will be merged to one if they are closer to each other than defined by "gtol". To split a line move the mouse pointer over the line and press the "s" key. Press "q" to quit the command. There is no undo but lines can be combined with the "qseq" command. Splitting of lines can also be done with "qlin". Surfaces and lines can be splitted with "qmsh" and "split".

It is also possible to measure distances between two pixels on the screen. Just press the key "w" on the positions of the two pixels. The distance is calculated in the scale of the displayed geometry.

9.107 qsur

```
'qsur' <name>(optional) RETURN
'w'|'a'|'b'|'l'|'i'|'r'|'1'-'9'|'g'|'q'|'u'|'s'|'S'|'h'|'c'
```

This keyword is used to create or change a surface (see also gsur). The user might specify a name in the command-line or by picking an existing surface with the "s" key.

If the surface is supposed to be meshed with unstructured triangles (element type tr3u or tr6u) it is sufficient to select all lines with the "l" key (lowercase "L"). This can be done either in the "all"-mode (press "a") or "individual"-mode (press "i") which is default. Then generate the surface by pressing the "g" key. Such surfaces allow holes in it. If the surface is not plane it will need to reference an embedded NURBS surface (see "nurs") or a shape (see "shpe"). Despite it is quite an exception it should be noted here that this kind of mesh might be extruded in the 3rd dimension by using the "swep" command to generate penta elements for the cfd solver.

For the definition of a regular meshable surface three to five edges must be defined. To create a surface move the mouse pointer over the first line of the first edge and select it with the "1" key (number "one"). If more than one line is necessary to define the first edge select the following ones one after the other with the "1" key. Each marked line is listed in the mother konsole. If all lines of the first edge are selected select the lines of the second edge by pressing the "2" key, then the third and eventually the fourth and fifth. The last selected line must match the first. All lines defining an edge will create a combined line (see lcmb) and this combined line will show up in the definition of the surface.

Then generate the surface by pressing the "g" key and you might continue with the next surface. By default the interior of the surface is defined according to Coons [14] which is a blended function. But it can also follow an embedded NURBS-surface or a shape. To relate a surface to an existing NURBS surface select another surface which already uses the desired NURBS with the "S" key (uppercase) or to reference the shape with the "h" key. Then select the target-surface with "s". The target surface is converted to a NURBS surface which is necessary to be meshable with unstructured triangles (tr3u, tr6u) and to be used to generate tet elements. The selected shape or nurbs will stay selected until cleared with 'c'. In case the NURBS or shape in question is not related to a surface it is necessary to use "qshp" to select first the NURBS or shape and then the surface which should reference them.

An existing surface might be completly redefined without destroying the definition of related bodies or other geometries. After selecting a surface with the "s" key the referenced lines might be replaced by selecting new ones. The previous selected lines are not longer referenced.

Replace assignments by the blended function with the "b" key. To quit the command use the "q" key or use "u" to undo the last action.

It is also possible to measure distances between two pixels on the screen. Just press the key "w" on the positions of the two pixels. The distance is calculated in the scale of the displayed geometry.

9.108 qtxt

```
'qtxt'  RETURN 'g'|'a'|'i'|'b'|'p'|'m'|'n'|'v'|'f'|'d'|'s'|'q'
```

This keyword is used to move node-attached texts (showing node-number and value, see figure 8) at certain positions in the drawing area or to manipulate them otherwhise. They are created with the key "g" while the mouse-pointer is over the node to which it should be attached. Attention: They are only visible after the command "plus nt all" was used (but they are immediatelly visible when the the commands "qenq" or "qadd" were used instead of "qtxt"). To move a text pick it at the lower-left corner with the key 'p' and place it with the key 'm' in the new location. To move it back to its node use 'b'. Delete them with 'd'. The node-nr and the value in the text can be switched on and off with the 'n' and 'v' key. The 'f' key will toggle the format of the value between "int", "float" and "exponent". See the command "font" on how to change the font-size. See "txt" for the batch mode command. Be aware that the "txt" command offers additional functionallity, especially a user defined text-string (description) can be added. These descriptions can be switched on and off with the 't' key and placed at a leading or trailing position with the 's' key.

To catch more than one entity with one stroke, type 'a' (all) before and create a rectangular picking area by pressing two times the 'r' key. Both strokes define opposite corners of the rectangle. To catch only the entity which is closest to the user type 'i' before.

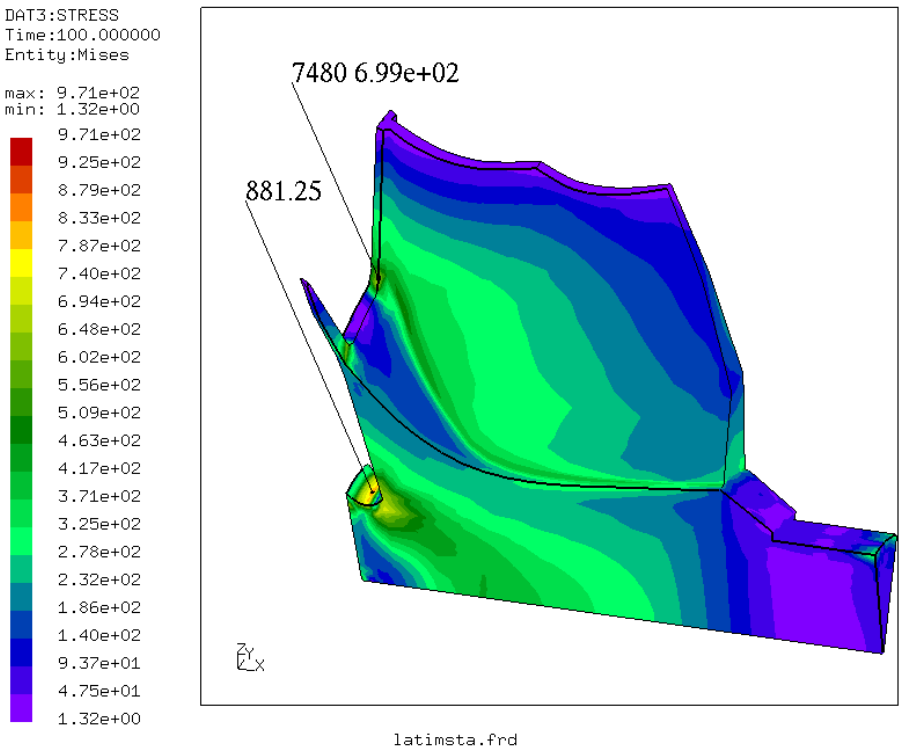


Figure 8: qtxt: Node attached texts at two locations, one with node-nr and value in exp-form and one just with his value in floating point form. The font with gives biggest numbers was used.

9.109 quit

```
'quit'
```

This keyword is used to terminate the program without save.

9.110 read

```
'read' [<geo-file> ['add']]|
       [<ccx-file> 'inp' ['add'|'ext'|'nom'|'new']]|
       [<openFoam-file> 'foam' ['add'|'ext'|'nom']]|
    [<result-file(frd)> ['add'|'ext'|'nom']|[<setname>]]|
       [<vtk-file> 'vtk' ['add'|'ext'|'nom']]|
       [<stl-file> ['add'|'ext'|'nom']]|
       [<list-file> '-n'|'-e'<column>]|
       [<text-file> 'stack']|
       [<edge-file> 'edg']|
```

```
[<netgen-file> 'ng' ['add'|'ext'|'nom'|'ndsb']]|
[<pixel-file> [<zoom>]|[<x_zoom> <y_zoom>]]
```

This keyword is used to read files or commands from a file. Most commands can be read from a file but not all of them. In general all basic commands to create geometry are understood and up to now this is the only way to read pre defined geometry during run-time. To read commands from a file like pnt, line, node, or seta and so on, type

read geo-file

this will eventually replace entities if their names were already in use. To prevent this, type

read geo-file add

this forces the program to change the names which are already in use to unused ones. Therefore no existing geometry will be overwritten. It is advisable to name command files which contain more than the basic geometry commands with a different name extention as '.fbd' since a subsequent 'save' would overwrite the command file with a file which contains just the basic geometry commands (it is a common practice to use '.fbl' instead).

If an ccx-input-file with loads, boundary-conditions and sets is read with a command like

read ccx-file inp

then the loads and boundary-conditions are stored in automatically created sets which start with a leading '+'. For example the ccx command '*BOUNDARY' will trigger the creation of the set '+bou'. The ccx sets are stored in cgx sets of the same name. The load-values are stored in Datasets.

The option "nom" (no-mesh) can be used to speed up the reading of frd or inp formatted files like:

read ccx-file inp nom

and/or

read result-file nom

This option suppresses the reading of nodes and elements and makes sense if the mesh exists and only the set-definitions and loads or results should be read.

If the parameter "add" is used,

read ccx-file inp add

then existing node- or element-numbers are not overwritten and the program choses new numbers. If the ccx-input-file ends with ".inp" the parameter "inp" might be omitted.

An openFoam file [9] can be read in a similar way

read CaseDirName foam

as an ccx input file. Sets will be created if defined in the openFoam file. Results can be used for mapping purposes. For further details see "How to map loads" in the appendix.

The user might read a result-file in frd format during runtime. If a mesh exists and should not be overwritten just add the parameter "add"

read result-file add

this forces the program to change the numbers which are already in use to unused ones. Existing datasets will be extended by the new node-numbers and their data. The option "ext"

read result-file ext

will also extend the existing datasets but in this case the nodes and elements are updated (modified). If no parameter follows then existing nodes and elements are updated and the new datasets will be appended to existing ones. Since the dependency checks are time consuming the user might scip them by using the 'ne' parameter. Then the existing mesh is deleted before the new one is read:

read result-file new

It is also possible to read files written in the result format (.frd) during runtime only to define sets of nodes or elements without changing the definitions of them. The command

read result-file setname

will create a set of the name setname and all nodes and elements listed in the file mesh.frd will be added to that set. But this will NOT create or modify the nodes and elements. All nodes and elements must exist and will not be changed.

In case a file written in an un-common format should be evaluated the user may read the file into the cgx stack memory:

read textfile stack

Then the user may loop over the stack using "while" and "valu" to read and

evaluate each record. Extracted data can be stored in new datasets (see “ds” and “node”). An example can be found in “User File Parser”.

A mesh in vtk format can be read. Still experimental, only some element types are supported

read file.vtk vtk

.

An ascii or binary stl file can be read with

read file.stl

The file stores just triangles which will be interpreted as unconnected triangle elements. The nodes of this mesh can be merged (“merg”) and the resulting connected elements can be used as a basis for a tet mesh (see “mesh” and “mids”).

If a file with the numbers of nodes or elements is not available in the result format (.frd) then a so called list-file could be read instead. In such a file either numbers of nodes or elements can be defined. The numbers found in a specified column is interpreted as a node- or element-number. If ”-n” is specified the numbers are interpreted as nodes and if ”-e” is specified the numbers are interpreted as elements. For example

read list-file -e3

will define a set storing names of elements from the third column of file list-file.txt. The created set-name is always the name of the file.

NETGEN Import

In case NETGEN is used for meshing then the edges which are generated by NETGEN can be included for modifications;

read edge-file edg

will create beam elements based on the defined edges. When the modifications are done, the beam elements can be exported in the NETGEN-edge format with the ”send setname stl” command and used for meshing in netgen (see also ”How to deal with CAD-geometry”). The netgen mesh can be imported with

`read netgen-file ng`

If the NETGEN (.vol) file contains solid elements, all shells and beams are only used to define surface sets of nodes and faces (+set<nr>). The shell elements and beams are deleted by default. If you want to be more selective about the

elements imported from netgen, you use the keyword ndsb (NoDeleteShellsAndBeams).

```
read myfile.vol ng ndsb
```

This keyword forces all the netgen elements to be imported. Be aware that the imported NETGEN element types (1D, 2D and 3D elements) are distributed in various sets for further operations (as well as being contained in set 'all'). For instance the set '+typ11' will contain all elements of type 11 (2-node beams). Please see "Element Types" for a key to the element numbers.

If a pixel-file in xwd-format is available it can be included as a background-picture. The user can then create geometry based on this picture:

read pixelfile 2 4

here the picture will be scaled by a factor of "2" in x- and "4" in y-direction. Delete it with "del pic". The picture can not be modified in cgx. Only scaling during reading (with either a global factor or separate scaling in x- and y-direction) is supported. Other modifications have to be made with external software.

The command parameters are writen to the "stack".

9.111 rep

```
'rep' <set>
```

This keyword is used to refresh entities of a certain set. This is done automatically but with one exception: The rendering of the interior definition of surfaces must be triggered manually with this command. The rendered surfaces can be displayed with "plot" or "plus" using the type "si".

9.112 rnam

```
'rnam' <set> <set>
```

This keyword is used to rename a set.

9.113 rot

```
'rot' ['n' <set>|'nodenr'] |
      [['u'|'d'|'r'|'l'|'c' <angle>]|->
       ['x'|'-x'|'y'|'-y'|'z'|'-z']]
```

This keyword is used to rotate the view-direction to the model. For example

rot u 10

will rotate the view-direction 10 degrees upwards. The meaning of the other letters is down d, right r, left l, clockwise c. The remaining letters will rotate the view into a pre-defined direction. For example the user will look in x direction after the command

rot x

and against the x direction with the command

rot -x

and so on.

The parameter 'n' is used to rotate the model in a way that the viewing direction is in normal direction at the location of a given node. Either a node-nr or a set with a node can be given:

rot n nodeset

Nevertheless the node might be obscured by other parts of the model. This can be checked with the "test" command.

9.114 save

```
'save'
```

This keyword is used to save the geometry (if any) to a file named as the input file but with the extension .fbd. If a file with that name exists already, then this file will be saved with the new file extension .fbb as a backup.

9.115 scal

```
'scal' ['s'|'v'|'d'] [<value>]
```

This keyword is used to scale the graphic presentation of values but no values of entities. The scaling of the value itself can be done with the command "ds". For example the command

scal s 0.5

will scale the range of values presented in the color plot by a factor of 0.5. The command

scal v 5.

will scale the length of vectors by a factor of 5. The command

scal d 2.

will scale the deformed shape by a factor of 2. Without a value it restores the default value. See also "Toggle Vector-Plot", "Toggle Add-Displacement", "Datasets".

9.116 send

```
'send' 'init'|
       <set>|[(dyn:)set#<nr>] 'fbd' ['c','e','f']| ->
             'stl'| ->
             'abq'|'ans'|'ast'|'dar'|'dyn'|'frd'|'gmp'|->
             'lst'|'nas'|'ng'|'pat'|'skv'|'tcg' <RETURN>|
             ['bin']|['dbin']|
             ['ds'<nr>[+] [e<nr>[','|'-'<nr>]..]]|
             ['tmf']|
             ['sta' <refspeed>]|
             ['crp' <timefact> <refspeed> <writefreq>]|
             ['comp']|
             ['mpc' [[<rotation>|'v'<node> <v1> <v2> <v3> ]|
                                ['n'<node>]]]|
             ['names']|
             ['raw']|
             ['spc' <dofs(1-6|t|p)> <value>]| ->
               [<dofs(1-6|t|p)> 'ds'<nr> e<nr>]| ->
               ['nor' <fac1> [<fac2> <fac3>]]| ->
               ['nor' 'ds'<nr> 'e'<nr>]|
             ['spcf' <dofs(1-6|t|p)> <value>]| ->
               [<dofs(1-6|t|p)> 'ds'<nr> 'e'<nr>]| ->
             ['slide' 'c'|'s'|'rx'| ->
               'ry'|'rz'|'tx'|'ty'|'tz']|
             ['trac' [<v1> <v2> <v3>]|['ds'<nr> 'e'<nr>(1.comp of a vec)] [+|-] ]|
             ['pres' [<value>]|['ds'<nr> 'e'<nr>] [+|-] ]|
             ['film' [[n<nodnr>][<temp>]|->
                   [[['ds'<nr>]|[sq<dsnr>-<dsnr>]] 'e'<nr>]->
               [[<coeff>]|->
               [['ds'<nr>]|[sq<dsnr>-<dsnr>] 'e'<nr>]] [+|-] ]|
             ['rad' [[<temp>|[cr<temp>]|['ds'<nr> 'e'<nr>]] ->
               [[<emis>|['ds'<nr> 'e'<nr>]] [+|-] ]|
             ['dflux' [[<load>]|['ds'<nr> 'e'<nr>]] [+|-] ]|
             ['mflow' [[<load>]|['ds'<nr> 'e'<nr>]] [+|-] ]|
             ['cflux' <load>]|
             ['force' <f1> <f2> <f3>]|
             ['quadlin']|
```

```
              ['sur' [+|-] ]
  'send' <set> 'dolfyn' [<type> <set>]
  'send' <set> ['duns'|'isaac'] [[<type>|'periodic'] <set>]
  'send' <set> ['foam'] [<type> <set>] | ->
              ['cyclic' <set> <set> 'tx'|'ty'|'tz'|'rx'|'ry'|'rz'|
              'cx'|'cy'|'cz'|<vx,vy,vz>]
  'send' <dep-set> <indep-set> 'nas'|'abq'|'ans' ['cycmpc'|'cycmpcf' ->
    'px'|'py'|'pz'|'tx'|'ty'|'tz'| ->
    'rx'|'ry'|'rz'|'cx'|'cy'|'cz'<+-segments> 'c'|'u'<NR>]| ->
    ['areampc' <dofs('t'|'p'|'1-6')> 'c'|'u'<Nr>]|'f'<value> ->
    ['areampc' 'slide']| ->
    ['areampc' 'presfit' ['s'<value>]]| ->
    ['gap' <vx> <vy> <vz> <tol>]
```

This keyword is used to send data to the file-system. In certain cases the written file-name contains given parameters to make them unique which is convenient when the user writes several files of the same type. But on the other hand this makes it sometimes hard to know in advance how the file would be named. Therefore the send command writes the file-name to the stack from which the user can derive the name if the "stack" was activated.

The following formats are known (but not all options for all formats are fully supported so far); The geometry-format (fbd)

send set fbd

or

send set fbd c

will complete the set downwards before writing.

Usually the point coordinates are written in exponential format but with option 'f' after 'fbd' the point coordinates will be written in long float (double) format:

send set fbd f

Useful when otherwhise a '0.' is not exactly zero. The command

send set fbd e

will write all model-edges as small lines running from node to node. A special case of a command file is written with

send init

named init.fbl which stores status variables (also written at the end of geometry

files). This can be used in a second or later session by reading it with the "read" command:

read init.fbl

The user might use the "menu" command to define two command lines for the writing and reading of the init file to a certain location. With this approach a certain orientation of the model can be quickly stored and restored.

A surface description with triangles (stl)can be written:

send all stl

This triangles are based on elements which were created by meshing surfaces or by automatically triangulated element-faces of all types of supported elements. If be2 elements are included in the mesh (meshed lines) a so called edge file for NETGEN will also be created if stl is written. NETGEN provides also a stand-alone mesher called ng_vol. The mesher can be found in the netgen subdirectory nglib. As for the stl-format triangles can be written for this mesher with:

send set ng

It has the advantage that the triangles are directly used to define tetras and not as with stl are only used to define the outer shape of the body.

The following mesh-related formats are known: Femview and CADfix (frd), Nastran (nas), Abaqus (abq), Ansys (ans), CodeAster (ast), Samcef (sam), Darwin (dar), ls-dyna (dyn), patran (pat, only sets), gagemap (gmp, only sets) and Tochnog (tcg) but only ccx (Abaqus) is fully supported. Good support is also provided for ansys, darwin and nastran. If no further parameter follows then just the definition of the nodes and elements will be written:

send all abq

will write the mesh in the format used by Abaqus and the CalculiX solver. If the parameter "ds" is provided followed by the dataset-nr

send set abq ds1

then the values stored in Dataset 1 are written. For 'frd'-format the 'ds' parameter w/o any further values forces cgx to write all datasets but w/o the internally calculated ones:

send all frd ds

If a "+" sign appears at the end of the 'ds' parameter

send all frd ds+

or

send all frd ds1+

then also the internally calculated values like vMises-stress are written. If the user needs to send more than one dataset but not all he can either just combine them (linux:cat) or use several send commands and read the resulting files one after the other in a new cgx session:

```
To write more than one dataset:
send all frd ds1
asgn mem free
sys mv all.frd ds1.frd
send all frd ds2
asgn mem free
sys mv all.frd ds2.frd

To combine the datasets in a new file using a new cgx session:
read ds1.frd
read ds2.frd
send all frd ds
```

The "asgn" command is optional and should be used if the model is big compared to the memory of the computer. Additionally the entities might be specified with the 'e' parameter followed by the numbers to write. The numbers have to be separated by a ',' sign or in case of a range by a '-' sign:

send all frd ds1 e9,25-30

The "bin" parameter has the same meaning as the "ds" parameter but in this case the result is written in the single precision binary form of the frd-format ('dbin' writes formally double precision but the data are single precision). It writes always all datasets. In case data should be written for the crack analysis tool Darwin

send all dar ds

will write all datasets. Since Darwin changed the format from version 7.1 on, it is possible to change the format with the parameter 'v7.1' as the last parameter (use 'v7.1' and the darwin conversion tool for newer versions). If the set is of the ordered type and includes nodes (see "qseq") then the data are written in tabular form for use in a 1D crack-prop Darwin-analysis.

In certain circumstances the user needs an easy solver independent format to write node- and element numbers. This is provided by the "lst" parameter:

send setname lst

will write the node- and element-numbers in lines of seven space separated rows.

The export of 'skv' input data for the quasi-normals is included:

send setname skv ds1

will write the node coordinates in y direction as 'r' coordinates, the x direction as 'z' coordinates, the values from the given dataset as the 'betaru' value, datasetnr+1 as the 'betazu' value, datasetnr+2 as the 'sigma' value and datasetnr+2 as the 'entropy' value.

Missing lower entities (nodes, points, lines etc.) will be added to the set before the set is written if the parameter "comp" follows. For example geometry like bodies

send set fbd comp

will be extended by surfaces, lines and points or

send set frd comp

will include all nodes used by the elements to the set elem and will then write the file.

The parameter "quadlin" forces the conversion from second order elements into single order elements were each element is subdivided into 8 single order elements. This takes place during writing and will not change the current state of the mesh in cgx. Up to now it is only available for the abq format:

send all abq quadlin

If the parameter "tmf" follows the definition of a solver format

send set abq tmf

then the mesh plus all temperatures with the necessary solver context for tmf calculations is written. If the parameter "sta" follows

send set abq sta 2900.

then the mesh plus all temperatures with the necessary solver context for a static calculations is written. Here the value for "speed" is a reference value for eventual numerical values in the TEXT-Block of the Datasets in the result file (frd-file, see "Nodal Results Block"). A scaling factor will be determined based on the reference value (here a ref.speed of 2900.) and the individual

TEXT-Block values (factor = speed**2 / refspeed**2). This factor is used in *AMPLITUDE commands which will also be written and will be used to scale static loads which are referenced in the *STEP data. If the parameter "crp" follows

send set abq crp 1. 2900. 1

then the mesh plus all temperatures with the necessary solver context for creep calculations is written. Here the value for "timefact" scales the time-values of the Datasets, "speed" is a reference value for scaling static loads (see option "sta") and writefreq limits the output to the result-file.

If the parameter "names" follows the solver-type then just the element- or node-numbers are written. This is useful for defining sets for the solvers. The command

send steel abq names

will write the node and element numbers included in "steel". This file could be used to define a set for material assignments, boundary conditions or loads. If all sets should be written at once, use:

send all abq nam

When using "abq" together with "names" or "sur" a set is defined before the data lines. If this is not desired then the "raw" parameter has to be used instead of "names" or "sur":

send nodeset abq raw

To write sets for ls-dyna it is necessary that an identification number is part of the setname. The '#' character is used to separate them:

send steel#1 dyn names

or

send surfaces#2 dyn sur

It is also possible to define and send some constraints to the file system. To constrain degrees of freedom (dof) of selected nodes use the parameter "spc" in combination with the numbers of the constrained dofs and optionally a forced deflection. For example:

send hinge abq spc 123

will constrain the translational degrees of freedom of the nodes in set "hinge". Or

send hinge abq spc 12356

will leave just one rotational degree of freedom unconstrained. And

send move-nodes abq spc 1 0.1

will move the nodes in set move-nodes by 0.1 in direction 1.

send move-nodes abq spc 1 ds1 e1

will use the values from dataset-nr 1 and the entity-nr 1 for the forced displacement in direction 1 and

send move-nodes abq spc nor 0.1

will force the nodes included in set move-nodes by 0.1 in the normal direction (normal to the element-faces). The components of the vectors at the node positions can be scaled individually (fac1-3)). The vector length can also be read from an entity of a dataset:

send move-nodes abq spc nor ds1 e4

A subsequent calculation with ccx will move the mesh accordingly. After solving the deformed mesh can be used as a new mesh for further calculations. With this procedure geometric variations are possible without manipulating the geometry (morphing).

For the ccx cfd finite volumen method there is a "spc" function available which works on faces instead on nodes (see *BOUNDARYF in the ccx manual).

send Inlet abq spcf 123

The degees of freedom 123 define velocities and not deflections. Its called "spcf" and works otherwhise accordingly to the "spc" command. For the *INITIAL CONDITIONS keyword of ccx you write the data for example with

send all abq ds123

were ds123 is the dataset-nr shown in front of datasets (not loadcases) in cgx. It can be a vector (velocity) or scalar (pressure, temperature).

The parameter "slide" in combination with one additional parameter will create equations that force the selected nodes to move only in a plane. If the additional parameter "s" is specified then the specified nodes will be attached to

their element faces. All nodes of the affected element-faces have to be selected. For example you type:

send sur abq slide s

and this should attach the node with the node-nr. 1 to a plane parallel to the surface of the corresponding element-face. This element uses the node 1, 2, 3, 4, 5, 6, 7 and 8 where 1, 2, 3 and 4 are on the surface of the mesh. The program will detect this element face and constrain the node only then if the nodes 2, 3 and 4 were also included in the set "sur"! The detected elements or faces of volume-elements will be assigned to the set. So it is possible to check the detected elements. Instead of providing a set with the necessary nodes, you might instead provide a set with shell-elements or/and faces of volume-elements. The nodes will be derived from them. In most cases this is the most convenient way.

A special case which works probably only in ccx (abaqus might give wrong results) is triggered by the "c" parameter. It behaves as for the "s" parameter but works for cycsym calcs:

send cycsur abq slide c

A new set with new nodes will be created with the name "<set>_COPY" which has to be used in the cycsym definition in ccx instead of "<set>". In addition equations are specified which connect the new nodes with the original ones in the normal direction of the element faces. In this way two faces used in a cyclic symmetry connection will slide at each other instead of beeing glued in all directions.

If nodes should slide radially to the y axis then use "ry":

send sur abq slide ry

or use "ty" if the nodes should move only tangential to the y axis. In both cases the axial movement is still permitted.

The parameter "sur" will write the surfaces of the mesh either in abaqus-format for the *SURFACE command, in ls-dyna or in frd-format as shell elements. For example

send top abq sur

will write the elements in set top together with the face-nr (Important: Only free surfaces of the mesh are regarded, internal faces are unknown and can not be identified). The front- or rear side of the face (pos or neg) is selected with the '+' or '-' parameter:

send surset abq sur -

will write all surfaces in set surset in abaqus format and all surfaces which have a potentially negative side will be written in that manner. To write surfaces for ls-dyna it is necessary that an identification number is part of the setname. The '#' character is used to separate them:

send surfaces#2 dyn sur

The parameter "pres" is used to assign pressure values to element faces. For example

send surf abq pres 0.05

will assign the pressure value 0.05 to all element-faces in set surf. If a dataset with nodal values is available it is also possible to use this values instead of using a uniform value for all faces. For example

send surf abq pres lc1 e1

will specify the dataset-nr 1 and the entity-nr 1 to be used instead of a uniform value. The front- or rear side of the face (pos or neg) is selected with the '+' or '-' parameter (it has to be the last parameter).

The parameter "trac" is used to assign a surface traction vector to element faces. For example

send surf abq trac 0.05 0.01 0.1

will assign the vector 0.05 0.01 0.1 to all element-faces in set surf. If a dataset with nodal values is available it is also possible to use this values instead of using uniform values for all faces. For example

send surf abq trac lc1 e1

will specify the dataset-nr 1 and the entity-nr 1,2,3 to be used instead of uniform values. The front- or rear side of the face (pos or neg) is selected with the '+' or '-' parameter (it has to be the last parameter).

The parameter "film" is used to assign free-stream temperatures and thermal heat coefficients to element faces. It works similar to the "pres" option. For example

send surf abq film 1200. 0.5

will assign the film temperature of 1200 and the coefficient 0.5 to all element-faces in set surf. If datasets with nodal values are available it is also possible to use this values instead of using a uniform value for all faces. For example

send surf abq film ds1 e1 0.5

will specify the dataset-nr 1 and the entity-nr 1 to be used for the temperature instead of a uniform value, or

send surf abq film ds1 e1 lc1 e2

will specify also the dataset-nr 1 with the entity-nr 2 to be used for the film-coefficients instead of a uniform value. The front- or rear side of the face (pos or neg) is selected with the '+' or '-' parameter (it has to be the last parameter). Instead of a single dataset-nr a range can be given:

send surf abq film sq1-9 e1 sq1-9 e2

and instead of giving e determined value for the temperature a node of a thermal network can be specified with:

send surf abq film n1277 ds1 e1

The parameter "rad" is used to assign a sink temperatures and the emissivity to element faces. It works similar to the "pres" option. For example

send surf abq rad 1200. 0.5

will assign the sink temperature of 1200 and the emissivity 0.5 to all element-faces in set surf. If datasets with nodal values are available it is also possible to use this values instead of using a uniform value for all faces. For example

send surf abq rad lc1 e1 1200.

will specify the dataset-nr 1 and the entity-nr 1 to be used for the temperature instead of a uniform value, or

send surf abq rad lc1 e1 lc1 e2

will specify also the dataset-nr 1 with the entity-nr 2 to be used for the emissivity instead of a uniform value. The front- or rear side of the face (pos or neg) is selected with the '+' or '-' parameter (it has to be the last parameter). If cavity radiation should be used then the command has to use the 'cr' attribute:

send surf abq rad cr1000. 0.7

Here 1000. is either the sink temperature, or, if the ccx parameter ENVNODE is active (*RADIATE,ENVNODE), the sink node.

The parameters "dflux" and "mflow" are used to assign an energy stream

or a mass-flow to element faces. They work similar to the "pres" option. For example

send surf abq dflux 0.5

will assign the flux of 0.5 to all element-faces in set surf. And

send surf abq mflow 0.5

will assign the mass-flow of 0.5 to all element-faces of the given set. If a dataset with nodal values is available it is also possible to use this values instead of using a uniform value for all faces. For example

send surf abq dflux lc1 e1

will specify the dataset-nr 1 and the entity-nr 1 to be used instead of a uniform value. The front- or rear side of the face (pos or neg) is selected with the '+' or '-' parameter (it has to be the last parameter).

The parameter "cflux" is used to assign an energy stream to nodes. For example

send surf abq cflux 0.5

will assign the flux of 0.5 to all nodes in set surf.

The parameter "force" is used to assign force values to nodes. For example

send nodes abq force 1. 20. 0.

will assign the specified forces to all nodes of the set nodes.

The parameter "mpc" is either used to create input for the user-subroutine umpc which forces all nodes from "set" to rotate by an average value specified with "rotation" around the vector v, or to create a rigid body. In case of a rotation the value has to be in degree were 90 degree is orthogonal. For example

send nodes abq mpc 4. 1. 0. 0.

will assign the nodes of the set "nodes" to the user-subroutine umpc and will force them to rotate by 4 degree around the x-axis. Two files are produced. One file with the equation and the reference node has to be included in the model-definition-section and the boundary-file in the step section. The reference node is used to apply the pre-defined rotation but can also be used to apply a moment by using the *CLOAD command. The number of the reference node is always one above the highest node number in the current mesh. You may use "node" to increase the last node number in the model.

In case of a rigid-body request there are two other parameters in combination with "mpc" available. The rotation value is replaced by a nodenr of an

independent node which will be created based on either provided coordinates

send nodes nas mpc v4711 1. 0. 0.

or by averaged coordinates based on the specified set (here “nodes”) which are interpreted as dependent nodes:

send nodes nas mpc n4711

In both cases an RBE2 (nastran) or *RIGID BODY,REF NODE=nodenr (ccx,abq) element will be created which connect all this nodes. See also "asgn" on how to predefine a thermal expansion coefficient “alfa” for this element (only nastran).

Another useful method are so called "cyclic symmetry" equations. These equations are used when just a section of a rotation-symmetric part like a disk is modeled. These equations force the two cutting planes of such a section to move exactly equal in the cylindrical system. If the coordinate system of the displacements for the solver is rectangular (xyz) then the syntax is:

send dep indep nas cycmpc rx12 c1

Here "dep" is the set containing the nodes of the dependent side. These nodes will be replaced by the solver with the independent nodes from the set "indep". In this case the equations will be written in the nastran format "nas" (in nastran called MPC). The parameter "rx12" defines the displacement system as rectangular "r", the rotational axis is "x" and the "12" defines the number of segments in 360 deg, therefore the angle of the segment is 360 deg /12. Attention: The sign of the number-of-segments must be negative if the angle between the independent side and the dependent side is negative. When the nr-of-segments is omitted the value is calculated individual for each node. The "c" triggers the correction of the position of the dependent nodes to a position defined by the angle of the segment (highly recommended), "u" would prevent the correction. The "1" will be the identifier for the equations if the format is nas (nastran). In case the format would be ans (ansys) then the "1" would be the number of the first equation. No number is required for abq (abaqus and calculix). If the coordinate system of the displacements is cylindrical (rtz) then the example would be:

send dep indep nas cycmpc cx12 c1

Only the "r" from "rx12" is changed to "c". A thermal connection is created with:

send dep indep nas cycmpc tx c

The thermal connection is triggered by the "t", a pressure connection with "p". See comments above for the single parameters. In case of the ccx cfd finite volumen method the syntax is similar but only the function name itself differs:

send dep indep abq cycmpcf rx12 c1

The function name for cfd elements is 'cycmpcf' instead of 'cycmpc'.

In addition it is possible to "glue" independent meshes together. For this purpose the dependent nodes are tied to independent elements by equations. Choose the finer mesh for the dependent side. The equations are based on the shape-functions of the element types. For example

send dep indep nas areampc 123 c1

will connect the nodes in the set dep to element-faces described by nodes included in the set indep. The set dep must contain all nodes which should be "glued" and the set indep should contain all nodes of the elements surfaces to which the dep nodes should be glued. The numbers "123" are the degrees of freedom which will be connected ("t" will create a thermal connection, "p" a press-fit connection). The "c" triggers the correction of the position of the dependent nodes to a position on the surface of the independent elements (highly recommended and default), "u" would prevent the correction, "f" forces the dependent node away from the independent face. Of course the mesh has to be written after the use of such a command, otherwhise the corrected node positions would not be regarded and the equations would lead to increased stiffness and decreased accuracy. The "1" will be the identifier for the equations if the format is nas (nastran). In case the format would be ans (ansys) then the "1" would be the number of the first equation. No number is required for abq (abaqus and calculix), see also "How to connect independent meshes".

There is also the "slide" option in combination with the "areampc" option: For example if the mesh of a turbine-blade and a disk should be connected with each other in a simple but realistic way then a sliding condition between this parts can be established. The command:

send dep indep abq areampc slide

will connect the nodes in the set dep to element-faces described by nodes included in the set indep but only in the direction perpendicular to a plane defined by either the related faces to nodes of the dep-set or to faces which are included in the dep set! This difference is important since in a situation were the nodes in the dep set do not fully define existing faces (e.g. line or point contact) the user can include in the dep set some faces which describe the desired sliding direction (most often some faces from the indep side). Therefore all dep-nodes and all indep-nodes must lie in the same plane and will slide in the same plane!

Another case is considered with the "presfit" option in combination with

the "areampc" option. For example if a cylindrical press fit should be simulated then a forced displacement between the two intersecting surfaces is necessary. This forces the dependent nodes to move to the independent face. Two modes are available:

send dep indep abq areampc presfit f

simulates sticking friction and with the option s

send dep indep abq areampc presfit s

works for sliding conditions. The user might request a certain value for the press fit if the overlapping of the mesh do not represent the necessary distance:

send dep indep abq areampc presfit s0.06

will move the dep-nodes 0.06 in the normal direction of the independent faces (works also with option f). Additional nodes are generated and can be used to request the reaction forces on the dependent nodes. They are stored in a set named N¡dep-set-name¿¡ind-set-name¿. Two files are produced. The one with the equations has to be included in the model-definition-section and the boundary-file in the step section.

To define so called "gap" elements and related control-commands: These elements will connect parts if they are closer as a certain distance. For example if the distance is zero (contact). The command:

send dep indep abq gap 1. 0. 0.

will connect the nodes in the set dep and indep with gap-elements but only if they match each other in the direction 1.

Special cases are the cfd-solvers Duns, Isaac and OpenFoam. The boundary patches are an integral part of the mesh. So it is necessary to specify all boundary patches when writing the mesh. Which means that all free surfaces of the mesh must be specified. This is an example for OpenFoam:

send all foam cyclic cyc1 cyc2 cx patch in patch out wall wall

will write the so called polyMesh description to the file-system. After the send command the set (all) with the mesh is specified, then the format (foam), then cyclic boundary conditions (cyclic cyc1 cyc2 cx) between set cyc1 and cyc2 of the axi-symmetric case around x (cx), then boundary conditions of type patch for set "in" (patch in), then boundary conditions of type patch for set "out" (patch out) and boundary conditions of type wall for set "wall" (wall wall).

The symmetric boundary-conditions (base-type: cyclic) can be axi-symmetric

(c) around x,y,z or rectangular (r) in direction of x,y,z. Only for OpenFoam and in the rectangular (Cartesian) case also a vector pointing in the direction of the symmetry can be specified (ie: 1.,1.,0.)

For dolfyn, duns and isaac the same syntax has to be used:

send all duns viscous-wall profil subsonic-inflow in subsonic-outflow out

send all isaac WALL profil FARFIELD far EXTRAPOLATE out

send all dolfyn INLE ingang OUTL uitgang ... etc. The parameters like 'viscous-wall', 'FARFIELD', 'INLE' are specific for the single cfd tools and are described in the documentation of these programs. The cgx knows a certain BC called 'periodic' to glue the left side to the right side of a mesh and so creating a periodic mesh (cascade). One set for each pair of lines or lcmb's (2D) or surfaces (3D) must be used. For example:

send all duns viscous-wall profil subsonic-inflow in subsonic-outflow out periodic cyc1 periodic cyc2 periodic cyc3

The cgx is able to write parts of the solver input file when certain values are defined (see "valu"). If they are defined the boundary conditions and gas properties will be written to the "duns.bou" or "isaac.dat" file. The following values can be provided for that purpose:

```
WallTemperature
temperature   // inlet
pressure  // outlet
stagnation-pressure  // inlet
stagnation-temperature  // inlet
u-velocity  // inlet
v-velocity  // inlet
ini-velocity-factor  // scales the initial velocity field
```

The user has to add missing commands before he can use the input files.

The initial flowfield values 'MACH' and 'ALPHA' for isaac is scaled by 'ini-velocity-factor' for better convergence if a 'JET' boundary condition is specified. In this case the inlet conditions are defined using the 'JET' conditions (the necessary inlet static pressure is calculated by cgx if the stagnation values are present). Further information is available in "Remarks Concerning Duns and Isaac". The velocity of the initial flowfield for duns is also scaled by 'ini-velocity-factor'.

Be aware that duns and isaac use block meshes which must be created using the set 'all'. So the use of set 'all' together with the parameter 'block' is mandatory when the block structure is needed! Nevertheless the user might only assign an element type to a sub set so that only a part of the geometry will be meshed (see "mesh").

9.117 seqa

```
'seqa' <seq> ['nod'|'pnt' <name> .. <=>]|
             ['afte'|'befo' <name> 'nod'|'pnt' <name>.. <=>]|
             ['end' 'nod'|'pnt' <name> .. <=>]
```

This keyword is used to create or redefine a set marked as a sequential set. This set is used for spline definitions (see line). With the command qlin such a sequential set is automatically created. To begin such a set type for example:

SEQA Q003 PNT P004 P005 P006 P00M P00N

The program will create or overwrite the set Q003. The command will continue in the next line if the sign "=" is found:

SEQA Q003 PNT P004 P005 P006 =
P007 P008 P009

The parameter AFTE will insert additional points after the first specified point in the existing sequence. The parameter BEFO will insert additional points before the first specified point and the parameter END will add additional points to a sequence.

9.118 seqc

```
'seqc' <set>
```

Converts an "lcmb into a spline. The lcmb will be deleted (not the referenced lines) and keeps the sum of divisions but sets the bias to 1.

9.119 seql

```
'seql' <set> <nr>
```

Makes splines from all sorts of lines. The nr of new created inner points is defined by the parameter nr. Also existing splines will be redefined.

9.120 seta

```
'seta' <set> ['!'|'n'|'e'|'p'|'l'|'c'|'s'|'b'|'L'|'S'|'v'|'se'| ->
             'sh'|'ld'<div> <[\]name|*chars* ..>] |
             ['n'|'e' <name> '-' <name> <steps>]
```

This keyword is used to create or redefine a set (see also qadd). All entities like points or bodies and so on must be stored at least in one set to be reachable.

The set "all" is created automatically at startup and will be open (see seto) all the time unless explicitly closed (see setc). To add points to the set "dummy" type:

seta dummy p p1 p2

This will add the points p1 and p2 to the set dummy. The following entities are known:

Nodes n, Elements e, Faces f, Points p, Lines l, Surfaces s, Bodies b, Nurb Lines L, Nurb Surfaces S, Values v, names of other sets se or shapes sh. If the entity of the specified type does not exits a set of that name is assumed and if existing then all it's entities of the specified type are appended:

seta set1 n set2

will append only the nodes in set2 to set1. If the type 'se' is used then the full content of set2 is appended. Wildcards (*) can be used to search for setnames of a certain expression.

The program will automatically determine the type of the entities if not specified, but then the names must be unique. More than one name can be specified. A minus sign between two numbers of nodes or elements specifies a range of entities with steps of "steps":

seta set1 n 1001 - 1100 12

If the '!' sign is specified instead of a setname then the program generates automatically sets with system defined setnames and stores entities in it. This can be used is to separate independent meshes and line-loops. The single independent meshes are then referenced by new setnames, for example:

seta ! all

will determine all separate (disjunct) meshes in set "all" and store them in sets called +CF<nr>. Figure 9.120) illustrates this function. Other entities have to be specified as an additional argument after the '!' sign:

seta ! l all

will determine all separate (disjunct) line-loops. Other entieties are not implemented so far.

Values "valu" can be used as arguments since they would be replaced by their content. It is necessary to mask a value with a leading '\' in cases were the value should not be replaces by its content:

seta set v \val1 \val2

will not substitute the val1 and val2 but add the values itself to the set.

Lines with a division higher than specified with

seta setname ld50

can be stored in a set. Here all lines with a division above 50 are stored in setname.

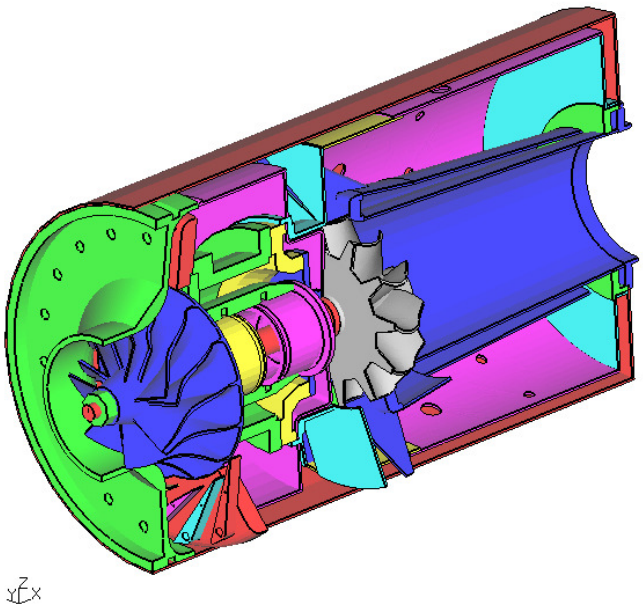


Figure 9: All disjunct meshes in the jet-engine are successive coloured

9.121 setc

`'setc' [<set>]`

This keyword is used to close an open set. Without parameter setc will close the at last opened set.

9.122 sete

```
'sete' <set> 'n'|'e'|'p'|'l'|'c'|'s'|'b'|'S'|'L'|'se'|'sh' ->
            'max'|'min'|'strict'
```

This keyword is used to enquire other sets which have entities in common or which have an identical content. Example:

sete blade p min

searches for sets who store at least the same points as "blade" (always the set "all"). The option "max" searches for sets whose entities are completely included in the specified set. The option "strict" searches for identical sets regarding the specified type of entity.

9.123 seti

```
'seti' <set> 'n'|'e'|'p'|'l'|'c'|'s'|'b'|'S'|'L'|'se'|'sh' ->
       <set> <set>..
```

This keyword is used to generate an intersection of certain sets (what do the specified sets have in common of the specified entity-type?). Example:

seti intersectSet p set1 set2 set3

generates a set intersectSet with points which are stored in each of the sets set1 set2 set3.

9.124 setl

```
'setl' <set> ['l']|'u'
```

This keyword is used to lock a set:

setl set

Or to unlock the set:

seto set u

A locked set will not store any additional entity and no entity will be removed. This can be used to prevent them from beeing unintendedly changed by using commands like "seto" or "swep".

9.125 seto

```
'seto' [<set>]
```

This keyword is used to enquire open sets:

seto

Or to mark a set as open:

seto set

All newly defined or redefined entities will be members of all open sets. See setc how to close an open set. An existing set which was locked with "setl" will be unlocked by this command.

9.126 setr

```
'setr' <set> 'n'|'e'|'p'|'l'|'la'|'ll'|'ls'|'ln'| ->
             's'|'b'|'L'|'S'|'v'|'se'|'sh'
             <name|*chars*> <name|*chars*> ..
```

This keyword is used to remove entities from a set (see also qrem. The entity will not be deleted. It is just not longer a member of that set. To remove entities from the set dummy type:

setr dummy p p1 p2

This will remove the points p1 and p2 from the set. The following entities are known: Nodes n, Elements e, Points p, Lines l, Surfaces s, Bodies b, Nurb Lines L, Nurb Surfaces S, Values v, other sets se and shapes sh. The program will automatically determine the type of the entities if not specified, but then the names must be unique. Wildcards (*) can be used to search for setnames of a certain expression.

The type of lines can be given with the second digit:

setr dummy ls all

This will remove only splines from set 'dummy'. Known are 'l' straight lines, 'a' arcs, 'n' nurbs, 's' splines.

9.127 shpe

```
'shpe' <name|!> ['pln' <P1> <P2> <P3>] |
                ['cyl' <P1> <P2> <R1>] |
                ['con' <P1> <P2> <R1> <R2>] |
                ['tor' <P1> <R1> <P2> <R2>] |
                ['sph'] <P1> <R1>]
```

This keyword is used to create a shape which can be used to define the interior of surfaces or to be used as a target for projections (see proj) or to split entities

(see split). REMARK: So far only the plane shape can be used for all kinds of projections and splitting. The other types can not be used for splitting and only the normal projection is implemented for them.

A plane shape is defined with the parameter "pln" followed by the names of three points:

shpe H001 pln P1 P2 P3

A cylinder can be defined by

shpe H001 cyl P1 P2 10.5

were 10.5 is the radius. A cone can be defined by

shpe H001 con P1 P2 10.5 2.

were 10.5 is the radius at point P1 and 2. at point P2. A torus can be defined by

shpe H001 tor P1 10.5 P2 2.0

were 10.5 is the radius at point P1. Point P2 lies on the torus axis and 2.0 is the radius of the tube. A sphere can be defined by

shpe H001 sph P1 10.5

were 10.5 is the radius at point P1. If automatic name generation is desired, then use "!" instead of a name. See also "qshp" for the mouse controlled definition of shapes. It should be mentioned that all shapes are actually nurbs and that internally all nurbs which are used by surfaces are actually referenced by a shape as an interface.

Remark: A further shape type exists which links a nurbs to a surface definition. Such a shape will be automatically generated when a nurbs is finished by the "END" parameter using the same name as the nurbs. This shapes will not be written to a file but using the prnt command will list them:

shpe N001 NURS N001

9.128 split

```
'split' [[<set> <set>] ['nos(eparation)'|'e']] |[<line> <point>]
```

This keyword is used to split lines, surfaces, bodies and elements stored in a first set by surfaces or shapes given in a second set. Also a single line can be splitted by a given point (see "pnt" how to generate a point on a line). It is advisable to split only elements or geometry. Otherwhise the performance is very poor.

So far only tetraeder elements (in the first set) can be splitted by surfaces,

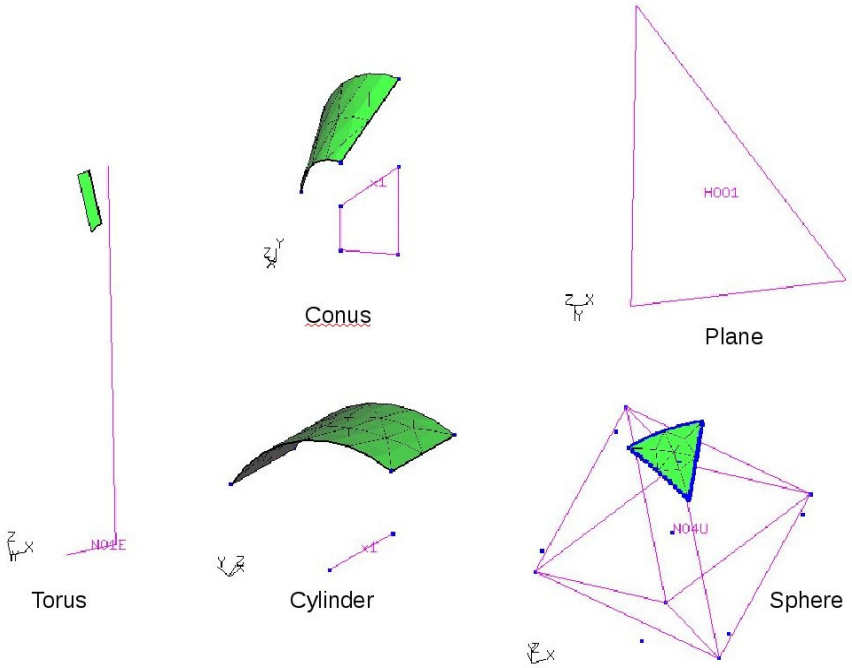


Figure 10: The shape types and its symbols together with an illustrating mesh

faces or shapes (referenced by the second set). The splitted tetraeder elements will be replaced by new tetraeder- and pentaeder elements. It can be used to cut out parts of the mesh to be either used directly in other models or which can be remeshed with tets based on an stl-file derived from them. Or it can be used to apply damages to a structure. See "How to map loads" in the appendix on how to map loads to the new created nodes from the original mesh. The use of parameter 'e' just stores the elements which would be splitted in the set '+eorig' but prevents the splitting of the elements:

split elset shape e

When bodies should be splitted then only one body per split command is currently possible. But only if the split would lead to exactly two parts. Splitting of several bodies at once will not create new bodies. Use several split commands instead. If unexpected results occur the user might use the option 'nos' to check the splitting of surfaces without the steps of separation and distribution to two bodies.

Sometimes the splitting of surfaces results in lines which do not look as

expected. The reason is usually a insufficient resolution of the triangles used for the representation of the surfaces. These triangles are generated or updated before the splitting happens and they are the basis for the splitting (the rep command does the same and the result can be viewed with "plot si all"). The user should increase the line division with qdiv and the inner resolution with elty or qmsh before he starts the next attempt (it is advisable to save the model before the split command is used).

When using the parameter 'noseparation' (first 3 chars are significant)

split surfset shape nos

the resulting surfaces will still be connected by the new lines created by the splitting. And they will eventually share a common NURBS. So if one of this surfaces is deleted it is necessary not to delete the related NURBS as well since otherwhise all remaining surfaces will lose their internal definition! Splitted surfaces can be combined again with "qmsh".

9.129 stack

```
'stack' on|off|free
```

Several functions which return values to the user might now place them also on a software-stack (used by area,dist,ds,enq,gtol,length,prnt,valu,volu and many more). The "valu" command is able to read this values from the stack and place them in variables. The 'free' parameter will empty the stack but does not open or close it.

See also "if" and "while".

9.130 steps

```
'steps' <value>
```

This keyword is used to define the number of different colors in the post-processor mode. The default is 21. The user might use this command to store his personal setting in the ".cgx" file in his home directory. See also "max".

9.131 stop

```
'stop'
```

Stops reading of the actual command file and gives the user the opportunity to work interactively until he resumes reading with "cont". It is possible to update the command file after the stop command.

9.132 subm

```
'subm' <submenuName> <name> <command>
```

This keyword is used to add a cgx command line to the menu like "menu". The difference is that this command creates or extends a submenu which allows the user to bundle related commands. This subm commands are usually stored in a ".cgx" file in the home directory to link them to the menu during startup.

9.133 surf

```
'surf' <name(char<9)>|'!' [<set>|<nurbs>]|<shape>] | ->
 [<nurbs>]|<shape>|'blend'] <line|lcmb> <line|lcmb> (etc. 3 to 5 times).
```

This keyword is used to define or redefine a surface. It is a more convenient way to define a surface than the command gsur. Either individual lines or a set of lines (including optionally one shape or nurs) can be specified. To be meshable with structured elements it requires 3, 4 or 5 edges (lines or lcmbs). To be meshable with tr3u or tr6u it requires either to be planar or to reference a shape or nurbs which can be provided also in a second call with this command. The mouse controlled way to define surfaces is to use the command qsur. If the name of the surface should be automatically generated, then just type "!" instead of a name. See also "gsur" which allows the assignment of a shape or "qshp" which allows the interactive assignment of a shape or nurbs to a surface.

9.134 swep

```
'swep' <set> <set> ['scal' <fx> <fy> <fz> <P0>|<div> [a] ]|
                   ['tra' <dx> <dy> <dz> <div> [a] ]|
                   ['rot' <p1> <p2> <alfa> <div> [a|&n] ]|
                   ['rot' 'x'|'y'|'z' <alfa> <div> [a|&n] ]|
                   ['rot' <p1> 'x'|'y'|'z' <alfa> <div> [a|&n] ]|
                   ['rad' <p1> <p2> <dr> <div> [a] ]|
                   ['rad' 'x'|'y'|'z'|'p'<pnt> <dr> <div> [a] ]|
                   ['rad' <p1> 'x'|'y'|'z' <dr> <div> [a] ]|
                   ['nor' <dr> <div> [a] ]|
                   ['mir' <P1> <P2> <div> [a] ]|
                   ['mir' 'x'|'y'|'z' <div> [a] ]|
                   ['mir' <P1> 'x'|'y'|'z' <div> [a] ]
```

This keyword is used to sweep entities into the next higher dimension. Sweeping a point will create a line, sweeping a line will create a surface and sweeping a surface will create a body. Shell elements will be expanded into Volume elements. The 'div' parameter defines how much elements will be created in the sweep direction. Existing results will be applied to the new nodes. Important: The "trfm" command must be used after the sweep operation and not before.

At first a copy of the first set (see seta and copy) will be created. The copy of the master-set is included in the second set. Then the connecting lines and surfaces are created and at last the bodies. The divisions of the new lines between set1 and set2 is specified with the parameter "div" or the default is used. Existing sets are extended by the copied entities if the last parameter includes the character "a" (append). Rotational sweeped lines create nurbs related surfaces if the last parameter includes the character "n". Several transformations are available. For example scal creates a scaled copy, the scaling factors fx, fy, fz could be chosen independently,

swep part1 part2 scal 2
swep part1 part2 scal 1 1 2 P0

tra will create a copy and will move it away by the vector dx, dy, dz and the optional parameter 'a' will assign the new entities to sets were the mother of each entity is included (see "setl" on how to lock and therefore exclude certain sets from that behaviour),

swep set1 set2 tra 10 20 30 a

nurbs related surfaces will be created if "a" is followed by "n" or a sole "n" is used in a rotational sweep,

swep set1 set2 rot 10 20 30 an

rot will create a copy and will move it around the axis defined by the points p1 and p2 by 'alfa' degrees (the connecting lines will be of type arc below 180 deg, above a spline),

swep set1 set2 rot p0 px 20.

or the axis of rotation is given by specifying one of the basis coordinate axes: swep set1 set2 rot x 20.

or just one point and a vector of rotation is given by specifying one of the basis coordinate axes: swep set1 set2 rot p1 x 20.

rad will create a copy and uses the same transformation options as 'rot' or will create a spherical section if just a single point is defined (be aware of the mandatory 'p' in front of the point name),

swep sphere1 sphere2 rad pP0 10.

nor will create a copy and will move it away in the direction of the averaged normal local vector. This requires information about the normal direction for each entity. Nodes will use associated element faces and geometric entities will

use the element faces, surfaces or shapes which must be stored with them in the set1. It should be noted that faces from volume elements stored in set1 will be used to generate shell elements. This elements will then be extruded in normal direction into 3D elements,

swep set1 set2 nor 1.2 6 a

mir will create a mirrored copy. The mirror-plane is placed normal to the direction running from P1 to P2 and placed at P2,

swep section1 section2 mir p1 p2

as with 'rot and 'rad' additional transformation options are available:

swep section1 section2 mir P1 x

places the mirror at P1 with its normal direction in 'x' direction

swep section1 section2 mir x

Places the mirror in the origin with its normal direction in 'x' direction.

9.135 sys

```
'sys' <shell-command parameters>
```

This keyword is used to issue any shell command (unix or dos shell). For example to move files created by the 'send' command to certain file names or/and locations and to start the analysis. On certain platforms cgx will not wait for the completition of the command if the '&' key was provided as the last argument of the command. Otherwhise cgx waits until the command was completed.

WARNING:

If you use CGX to open an untrusted .fbd file downloaded form the internet, it might delete all your files or do whatever it wants as long as cgx has the necessary rights. Even an honest but carelessly written .fbd file could be destructive if it makes incorrect assumptions about the locations of files. For example, by clearing the contents of a directory to clean up, it might delete important files on someone else's computer.

Therefore before you open a foreign command file (usually with the ending .fbd or .fbl) scan for "sys" and evaluate the command line.

To activate the "sys" command permanently add

```
'allow_sys'
```

in your configuration file ('.cgx' in your home directory).

9.136 test

```
'test' ['d'|'v'|'n'|'e'|'p'|'l'|'c'|'s'|'b'|->
        'S'|'L'|'se'|'sh' <entity-name>] | ->
       ['file' <file> [<">string<">]]  | ->
       ['i' <set1> <set2>] | ->
       ['o' <set>|<node>]
```

A boolean opperation. Usually tests the existence of a given entity (watch the '\' which prevents the value beeing replaced by its content!):

test v \value

And it checks if and which nodes included in set1 are inside the elements included in set2:

test i nodeset elemset

It returns TRUE or FALSE and the amount of nodes inside the elements of set2 and stores the nodes in the special set "-included".

Or it tests the observability 'o' of a certain node under the given orientation of the model. The node can be given as a member of a set or directly by its node-nr.

The existence of a file or the existence of a certain string in a given file can be checked:

test file filename.txt "HELLO"

The command returns TRUE or FALSE and writes it to the "stack".

9.137 thrs

```
'thrs' <set> <value> 'h'|'l'|'o' ['t']
```

Compiles all nodes from the given set which have values of the actual dataset above (h) or below (l) the given value (stored in set '+THRS'). The related elements are collected as well. All unconnected element-clusters from that set together with their related nodes are stored in separate sets (+grp1..+grpn). This sets are displayed in a certain mode were only the dots are visible (see "view"):

thrs all 600. h

When the parameter 't' is used the local nodes referencing maximum or minimum values are stored in additional sets (+grpN1..+grpNn) and displayed in addition:

thrs all 600. h t

This node attached texts can be manipulated with “qtxt” or with “txt”.

To display the full model again in the filled mode:

thrs o

9.138 tra

```
'tra' 'f'|'u'|'d'|'l'|'r' <relative-distance>
```

This keyword is used to move the model in the window. For example

tra u .1

will move the model 0.1 times the model dimensions upwards. The meaning of the other letters is forward f, down d, right r, left l.

9.139 trfm

```
'trfm' 'rec'|'cyl' ['x'|'y'|'z'] ->
     [<first-Dataset-Nr> [<last-Dataset-Nr>]]
```

Changes dataset entities from one coordinate system to another. The option 'cyl' transforms the global results to cylindrical and 'rec' from cylindrical to global cartesian. In both cases the axis of the cylindrical system must be provided. Optionally the first and the last dataset of a range of a unique type can be specified. The current dataset is selected if no dataset is specified. In any case a dataset parameter will be created which stores the type of the applied transformation. It will show up after re-selecting the dataset in the menu entry 'dataset->entity->parameter' and will be written by the "send" command if the frd-format is used (see also "ds" and Parameter Header Record).

The transformation into the cylindrical system takes place in a way that tensors and vectors are transformed into a new local cartesian system which is alligned with the directions of a true cylindrical system. In this way the dimensions are maintained (for example the displacement in angular direction is not transformed into an angle but into a displacement in tangential direction).

The transformation from a cylindrical into a cartesian system works accordingly. Therefore successive "cyl" and "rec" commands are permitted. This command sequence can be used to rotate the model with its datasets in the correct way which means that all results are also rotated.

Example; Choose the desired dataset (and an entity) with the menu or the 'ds' command. Then type

trfm cyl z

to transform the dataset from a rectangular system into a cylindrical around the global z axis. Type

trfm rec z

to transform from cylindrical (which exists after the first call) to the rectangular system (which re-produces the original values).

To transform several datasets of the same type (!) at once:

trfm rec z 1 10000

This command transforms all datasets starting with the first to the last if the last dataset has a number below 10001 (but only the ones of the same type as the 1st!).

9.140 txt

```
'txt' '!'|<set>|<node> ->
      ['n'|&'v'|&'t'|&'e'|&'f'|&'i'|&'s'|&'l'|&'r'] ->
      [<dx> <dy>] [<"description">]
```

Creates node-attached texts, see figure 8). The texts are generated but not displayed. The commands "plot" and "plus" have to be used for that purpose (the color 'n' is recommended since it is good visible over the white background as well as over the colors of most colormaps). The texts can be deleted with "del". See also "qtxt" for the interactive command.

The texts comprise of a node-nr and the value and an user defined description. The node-nr can be deselected by providing the key 'n', the value can be deselected with key 'v' and the description with key 't'. The exponential format of the value can be selected with 'e', float with 'f' and integer with 'i'. The left bound position can be defined with 'l', right bound with 'r'. All this flags can be combined.

The texts are attached to the nodes stored in a set;

txt nodset

or to a single node:

txt nodenr

If the ! character is used instead of a setname or a node number then a new set with the name “-txt” will be created and all nodes with attached texts will be stored there and the texts will be redefined:

txt !

Regard the following examples:

txt nodset fn

deselects the node-nr and selects the float format for the value.

The text is placed left bound if using the parameter ’l’:

txt nodset l

An eventually given vector dx,dy is scaled to the left bound position:

txt nodset l 0.1 0.5

Using an infinite gradient in the left- or right bound mode

txt nodset l 0. 0.5

will place the texts in the upper corner.

If not right or left bound the optional dx,dy values define an offset (normalized coordinates) to the actual node position on the screen where the origin is in the lower left corner:

txt nodset 0.1 0.2

The offset “0 0” will bring them back to the node position.

The description must be given in quotation marks:

txt nodenr “hello world”

Per default the description is placed at the trailing position. The key ’s’ shifts it to the front. Be aware that descriptions are persistend and can only be overwritten. To delete descriptions they have to be overwritten with an empty string:

txt nodenr “”

9.141 typs

```
'typs'
```

The result format (frd) allows to assign element types. One element can only belong to one type. This command creates sets and stores all elements of a certain type in a certain set called "+typ[nr]". See "Element Definition Block" how this applies to the result format (frd).

9.142 ucut

```
'ucut'
```

If a section through the mesh was created with the "cut" or "qcut" command then this command will delete the cut and display the un-cutted structure.

9.143 ulin

```
'ulin' <string>
```

This keyword is used to define an underline. This commend will show up in the menu area of the main window below the file name. The filename can be overwritten with "capt".

9.144 val

```
'val' (The parameters are the same as for 'valu'->
       except that masking with '\' is supported)
```

This command has the same functionality and accepts the same parameters as "valu" with one exception: The cgx command parser will substitute the parameters by previously defined values before the 'val' command itself is executed. The substitution can be suppressed with a leading '\' before a parameter. During execution it will again scan each parameter for expressions which match the name of a value and will replace the parameter by the content of the value. This way two steps of substitutions of nested values are possible:

```
valu a b
valu b 1.
# with valu only one substitution step is performed:
valu c a
prnt v c
-> b
# with val two substitution steps are performed:
del v \c
val c a
prnt v c
-> 1.
```

Please study the example "Data storage in a user dataset".

9.145 valu

```
'valu' <[!]name> [['push' [<splitkey>]]|['pop' [nr]]] | ->
      [<value> ['?' [<\"string\">]] | ->
      ['&'|'*'|'/'|'+'|'-'|'abs'|'max'|'min'|'pow'|'log'|->
       'log10'|'sqr'|'sin'|'cos'|'tan'|'asin'|'acos'|'atan'| ->
       'int'|'float'|'exp' [name|<const> name|<const>]] ]
```

WARNING: With that command the meaning of a command can be changed and unintended effects are possible. For example if the character 'l' is used as a value it is not longer possible to use the 'plot' command to display lines without masking the 'l'. So it is a good idea to use only names using several characters and avoid the names of existing commands.

This command generates an entity (called value) which basically stores a string of characters. Most characters are valid but no white-spaces are accepted from the command line. The command allows simple calculations and string operations. It is able to read from- and write to the stack. The cgx command parser will scan each parameter of each command for expressions which match the name of a value and will replace the parameter by the content of the value. After that the command is executed. For example

pnt P0 xvalue 0. 0.

uses the value 'xvalue'. If the user has previously defined the value with:

valu xvalue 1.24

then the command-parser will replace 'xvalue' by '1.24' in the 'pnt' command.

For convenience this general substitution works for all commands except the 'valu' command! The command parser will not scan the parameters of the 'valu' command and will not substitute them. Instead this substitution is performed by the 'valu' command itself. The command does not treat its own name as a value and will not substitute it by previously defined values. Therefore nested levels of 'values' are not solved. However if this functionality is needed the "val" command can be used.

ATTENTION: To suppress the substitution of a certain value it has to be masked by a leading '\'. For example the command:

del v \xvalue

will delete the value xvalue itself. Without the masking xvalue would be replaced by its content '1.24' and since no value named '1.24' exists, nothing will happen.

The content of a value can be defined by the user

valu arg1 1.24

or derived from the stack previously filled by a command

valu arg1 pop

if the "stack" was activated. A certain number on the stack can be addressed with

valu arg1 pop 2

were the '2' means that 2 successive 'pops' are executed. In this way the second value on the stack can be loaded at once.

Values can be added to a set

seta valset v all

and can then be deleted by zapping "zap" the set storing the values.

Values can be written to the stack when the 'push' parameter is used:

valu arg1 push

A white-space separated string stored in a "valu" will be written to the stack in separate sub-strings. With this method single pieces of a white-space separated string can be splitted and stored in separate values:

stack on

opens the stack

valu complicatedString push

splits the string stored in 'complicatedString' and writes the single pieces to the stack. The command

valu subString pop 3

loads the 3rd substring into the varible 'subString'. Other splitting characters can be used when this character follows the push key-word:

valu string push .

splits the string at each occurence of "." and writes the pieces to the stack.

The command is able to perform simple calculations like

valu result * arg1 arg2

'result' will store the product from 'arg1' and 'arg2'. The two arguments arg1 and arg2 may be other 'values' or constant numbers. That means that a direct multiplication of two numbers or a value with a number is possible. During the calulation the strings are converted to double precision numbers and the result is stored as a string representing an exponential number. The 'int', 'float' and 'exp' convert between integer, floating point and exponential format:

valu result int result

The '?' operator is used to request user input:

valu string ? "user message:"

The command presents the user message (between apostrophes) and waits for user input. Please type into the terminal. Usually this requires to leave the graphics window and click into the terminal from which cgx was started.

The '&' operator is used to concatenate two strings:

valu string3 & string1 string2

The string1 and string2 might be values or constant strings.

The values are written to the fbd file unless its name starts with a '!'.

The sections "How to write values to a file", "How to process results" and "How to generate a user dataset" explain more about the use of values.

9.146 view

```
'view' 'fill'|'line'|'point' [<value>]|'cl' 'off'|
       'edge' [<value>|'off']|'elem' ['off']|'surf'|'volu'|
       'front'|'back'|'vec' ['off']|'disp' ['off'|'keep']|
       'bg' ['w'|'k']|'sh' ['off']|'ill' ['off']|'rul' ['off'|<string>]
```

Command to control the graphic output. This command is intended for batch-mode. See also "Viewing" for the menu controlled functions.

- "cl" The command line is shown in the graphic's window.
- "fill" Element-faces are filled
- "line" Elements are displayed as fireframes
- "point" Element-edges are displayed as points with a pixel-width of 'value'
- "edge" triggers the display of the model edges with a pixel-width of 'value'
- "elem" triggers the display of the element edges. They are deselected with the additional parameter "off".

- "surf" and "volu" are used to display the structure either only by it's outer skin (surf) or by drawing all elements (volu).
- "front" and "back" define which side of the structure should be drawn. Either the side which faces the user or the back-side. If the back-side is displayed then internal structures are visible.
- "vec" triggers the vector mode. All vector-entities like displacements are displayed with arrows pointing in the direction of the vector and with a length proportional to the value of the vector. See "Toggle Vector-Plot" for a detailed description of the equivalent menu-function.
- "disp" will show the deformed structure based on a formerly selected displacement dataset (no entity must be selected). See "Toggle Add-Displacement" for a detailed description of the equivalent menu-function. Switched off with "off". Or add the displacements permanently with "keep". Now a second displacement dataset could be added. Be aware that the scaling of the displacement must be done after the 'view disp' command and before the 'view disp keep' command.
- "bg" Without second parameter toggles the background colour. The second parameter 'w' forces white while the parameter 'k' forces black as background colour.
- "sh" Shaded results are shown. Switched off with "off".
- "ill" Illuminale the backface of the elements. Switched off with "off".
- "rul" triggers the display of a ruler bar. Switched off with "off". A string containing the unit can be given: "view rul mm".

9.147 volu

```
'volu' <set>
```

This keyword is used to calculate the volume and the center of gravity of a set of volume-elements. If an ccx-input file with density data was read then the mass will be also calculated. If a 'dataset' is active then an averaged value is calculated.

The command writes to the "stack".

9.148 while

```
'while' <value>|<const> 'eq'|'ne'|'=='|'!='|'<'|'>' <value>|<const>
```

A command to compare two values ('valu' or constant numbers). If the compare is True the following commands are executed until the 'endwhile' command is found. This procedure is repeated until the compare is False.

while arg1 == arg2

will repeat the commands between 'while' and 'endwhile' until the numerical value stored in 'arg1' is not equal to the numerical value stored in value 'arg2'. The values are locally converted to 'float' format for the numerical comparison. The 'eq' and 'ne' compare strings and should not be used for numerical values since no conversion to a common format is done. Two strings are equal if they have the same length and all characters are equal.

See also "if", "valu", "stack" and "How to run cgx in batch mode".

9.149 wpos

```
'wpos' <xp> <yp>
```

Positions the window on the screen were xp and yp are the coordinates in pixel relative to the left upper corner. This command takes no effect during reading and execution of a batch file. It will be executed in the glut event loop (the glut library [2] for window management and event handling). The user might store his personal start-up location in the ".cgx" file in his home directory.

9.150 wsize

```
'wsize' [RETURN]|'f']|[<xp> <yp>]
```

Size of the window were xp and yp are the size in pixel. Without argument the size is the initial size and with the argument 'f' (fullsize) the screen resolution is used. This command takes ONLY effect during reading and execution of a batch file IF it is written in the very first line. Otherwhise it will be executed after reading of the batch file in the glut event loop (the glut library [2] for window management and event handling) which means that the program has to go into the interactive mode to take effect. The user might store his personal start-up window size in the ".cgx" file in his home directory.

9.151 zap

```
'zap'  <set>
```

This keyword is used to delete all entities of a set and the set itself. All depending entities will be deleted as well.

9.152 zoom

```
'zoom' [<scale>]|[<p1x> <p1y> <p2x> <p2y>]
```

This keyword is used to scale the model in the window. For example

zoom 2

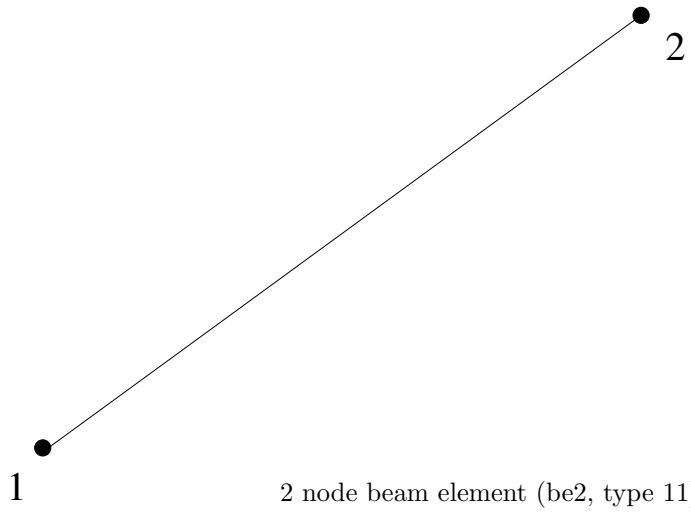
will increase the size of the representation of the model by a factor of 2. A certain region of the model can be specified with two corner points of an imaginary rectangle. The coordinates are relative to the graphic-window which has its origin at the left/lower corner and as a fraction of the edge-lengths. For example

zoom 0. 0. 0.5 0.5

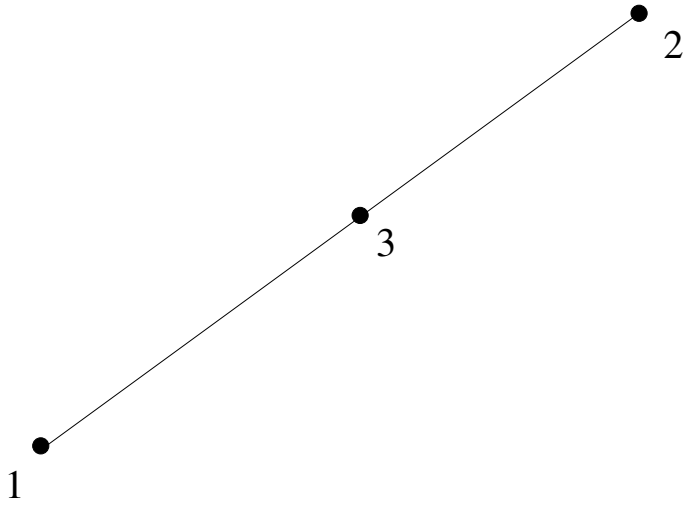
will display the third quadrant of the window scaled by a factor of 2.

10 Element Types

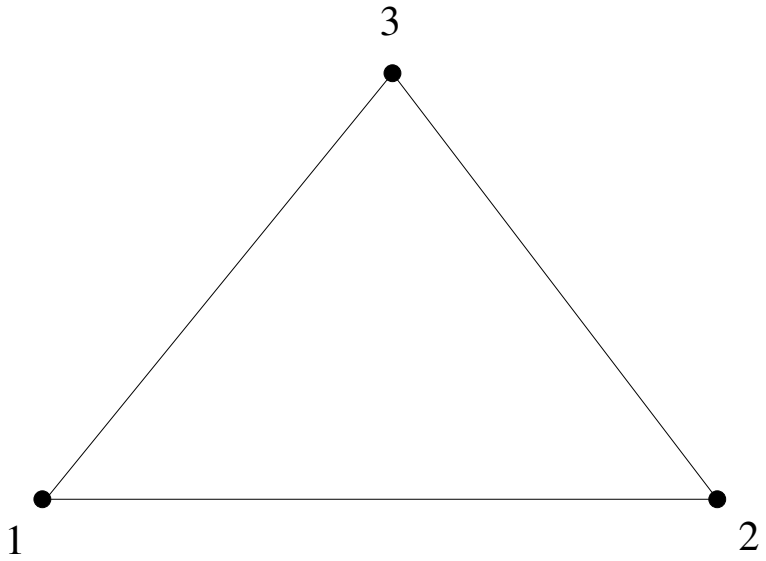
Node numbering of the elements and the type numbers used in the Result Format (frd-file). The solvers might use different node-numbering rules.



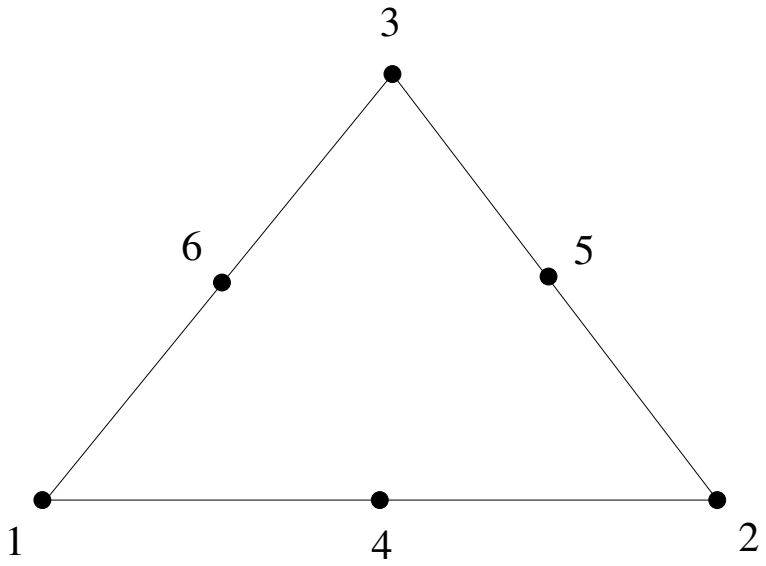
2 node beam element (be2, type 11)



3 node beam element (be3, type 12)



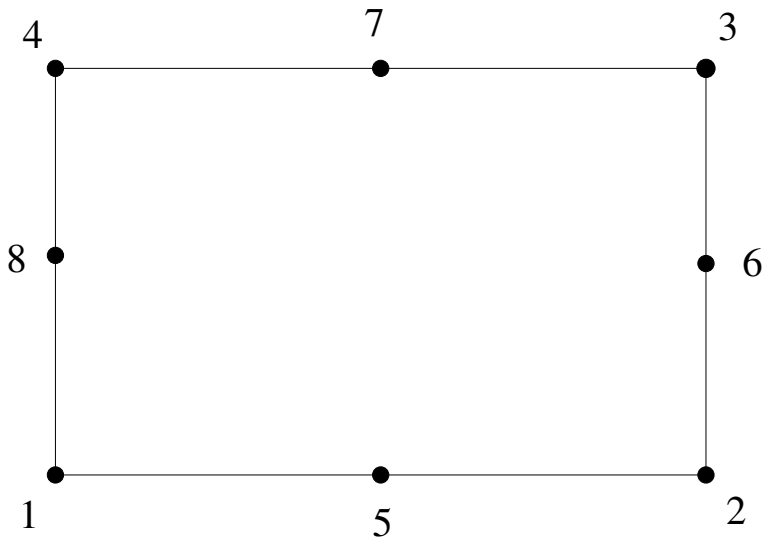
3 node shell element (tr3, tr3u, type 7)



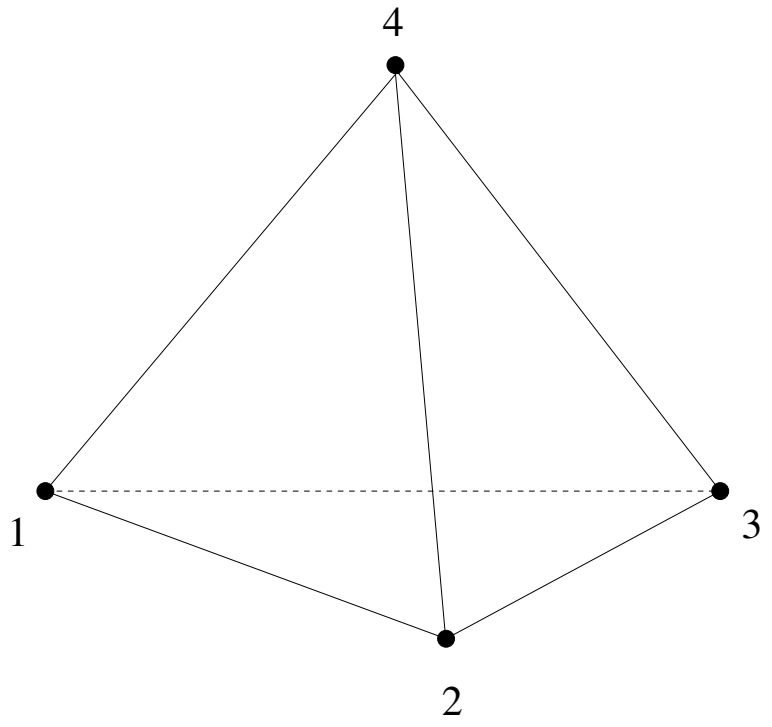
6 node shell element (tr6, type 8)



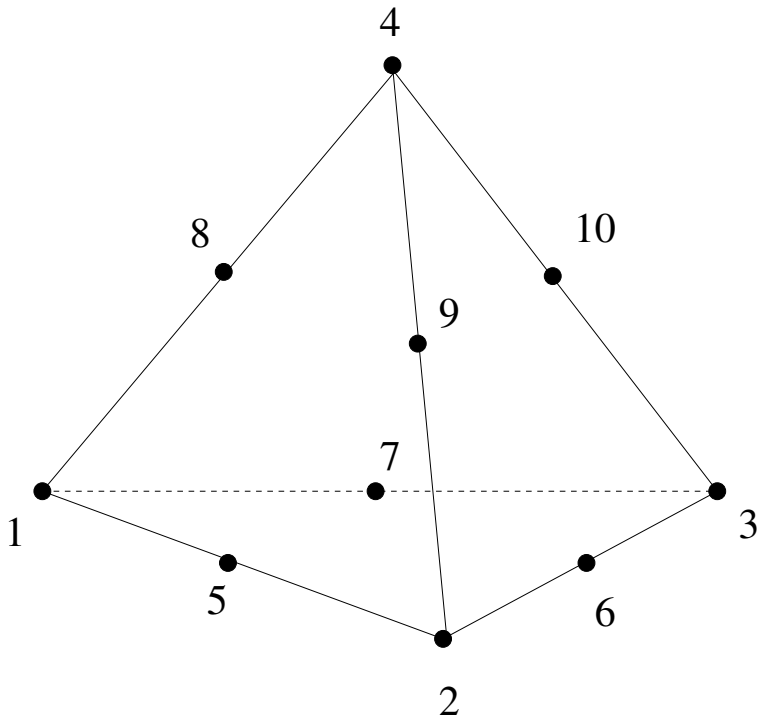
4 node shell element (qu4, type 9)



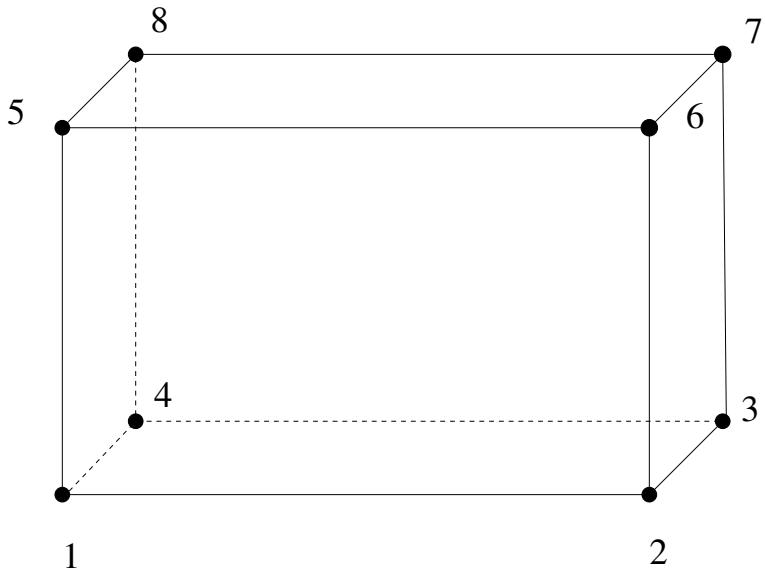
8 node shell element (qu8, type 10)



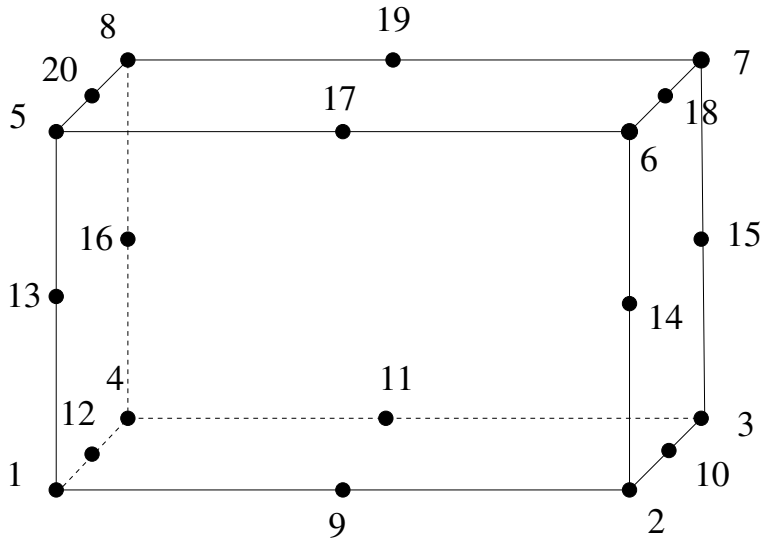
4 node tet element (type 3)



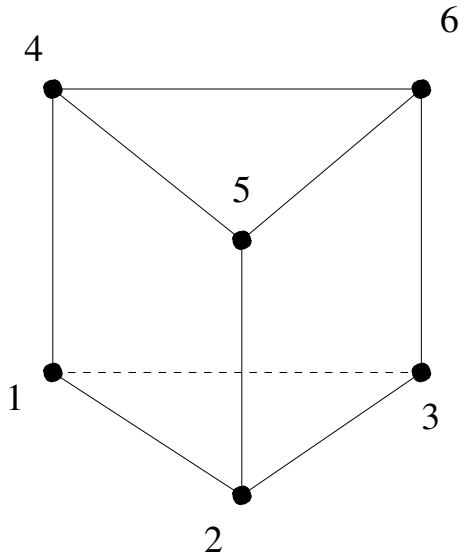
10 node tet element (type 6)



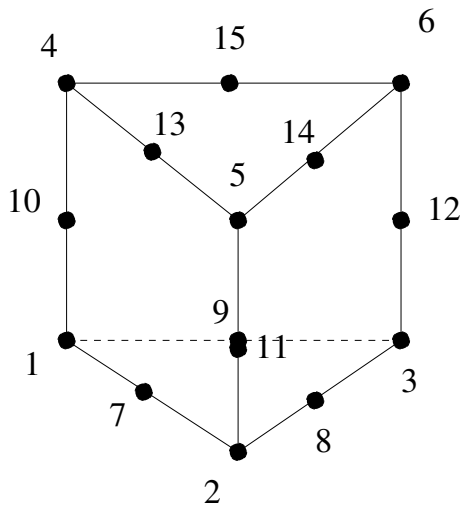
8 node brick element (he8, type 1)



20 node brick element (he20, type 4)



6 node penta element (pe6, type 2)



15 node penta element (pe15, type 5)

11 Result Format

Listing of the mesh- and the nodal results format. The data are stored in fixed format. Node-, element-definitions and results might be in ascii or binary coding. The ascii format is able to store element- and node-numbers up to '99999' in the short form or up to '9999999999' in the long form. An example for the short form is shown below:

```
>    1Ctest             !''1C'' defines a new calc of name ''test''
>    1UDATE  26.01.2000 !''1U'' stores user job-informations
>    2C                 !''2C'' starts a block of node coordinates
> -1    1 0.00000E+00 0.00000E+00 0.00000E+00 ! 1. node
> -1    2 0.10000E+01 0.00000E+00 0.00000E+00 ! 2. node ....
> -3                    !end of the current block
>    3C                 !''3C'' starts a block of elem definitions
> -1    1    4    0    0!first elem, type of that elem is 4 (he20)
> -2    1    2    3    4   13   14   15   16    5    6    7    8 ..
> -2   12   17   18   19   20 !twenty nodes defining that element
> -1 ...
> -2 ...
> -2 ...
> -3                    !end of the current block
>    1PHID      10      !defines a parameter HID value 10
>  100CL101            !''100C'' starts a user defined result block
> -4  DISP        3    1         !Attribute Header Record (Dataset)
> -5  D1          1    2    1    0 !Component Def. Record  (Entity)
> -5  D2          1    2    2    0
> -5  D3          1    2    3    0
> -1    1 0.00000E+00 1.00000E+00 1.00000E+00  !Nodal Values
```

```
> -1    2 1.00000E+00 0.00000E+00 0.00000E+00
> -3                      !end of the current block
> 9999                    !end of data
```

The binary format applys only for data lines and the “end of the current block” line (-3), which is omitted. All other lines are the same.

11.1 Model Header Record

To access the data use “prnt info”.

```
Purpose: Defines the name of the model

Format:(1X,'   1','C',A6)
Values: KEY, CODE, NAME
```

11.2 User Header Record

To access the data use “prnt usr”. To manipulate data use “asgn usr”.

```
Purpose: Stores additional user informations regarding the job
         ie. user id, creation date, model informations

Format:(1X,'   1','U',A66)
Values: KEY, CODE, STRING
```

11.3 Nodal Point Coordinate Block

Nodes can be written with the “node” command.

```
Purpose: Defines the nodal coordinates

1. Record:
Format:(1X,'   2','C',18X,I12,37X,I1)
Values: KEY, CODE,NUMNOD, FORMAT
Where: KEY    = 2
       CODE   = C
       NUMNOD = Number of nodes in this block
       FORMAT = Format indicator
                0  short format
                1  long format
                2  binary format, coordinates float
                3  binary format, coordinates double

Following records (ascci, FORMAT=0 | 1):
 Short Format:(1X,'-1',I5,3E12.5)
```

```
 Long Format:(1X,'-1',I10,3E12.5)
 Values: KEY, NODE, X,Y,Z
 Where: KEY    = -1
        NODE   = node number
        X..    = coordinates

Following records (binary, FORMAT=2):
 Format:(int NCOMPS*float)
 int and float are ansi-c data-types
 Values:  NODE, X,Y,Z
 Where:
        NODE   = node number
        X..    = coordinates

Following records (binary, FORMAT=3):
 Format:(int NCOMPS*double)
 int and double are ansi-c data-types
 Values:  NODE, X,Y,Z
 Where:
        NODE   = node number
        X..    = coordinates

Last Record (only FORMAT=0&1 (ascii), omitted for FORMAT=2&3):
Format:(1X,'-3')
Values: KEY
```

11.4 Element Definition Block

Elements can be written with the "elem" command.

```
Purpose: Defines the topology of the elements

1. Record:
Format:(1X,'   3','C',18X,I12,37X,I1)
Values: KEY, CODE,NUMELEM, FORMAT
Where: KEY    = 3
       CODE   = C
       NUMELEM= Number of elements in this block
       FORMAT = Format indicator
                0  short format
                1  long format
                2  binary format

Following records (ascci, FORMAT=0 | 1):
 The following block of records must be repeated for each element:
  The first record initializes an element definition:
```

```
  Short Format:(1X,'-1',I5,3I5)
  Long Format:(1X,'-1',I10,3I5)
  Values: KEY, ELEMENT, TYPE, GROUP, MATERIAL
 Where: KEY    = -1
       ELEMENT = element number
       TYPE    = element type, see section ''Element Types''
                 and command ''typs''
       GROUP   = element group number, see command ''grps''
       MATERIAL= element material number, see command ''mats''.

  Then the nodes in the correct order have to follow:
  Short Format:(1X,'-2',15I5)
  Long Format:(1X,'-2',10I10)
  Values: KEY,NODE,NODE,NODE,NODE, ...
 Where: KEY    = -2
       NODE    = node number

Additional lines must follow if more nodes are used.

Following records (binary, FORMAT=2):
 Format:(4*int nodes*int)
 int and float are ansi-c data-types
 Values:  ELEMENT, TYPE, GROUP, MATERIAL,NODE,NODE,NODE,NODE, ...
 Where:
       ELEMENT= element number
       TYPE    = element type, see section ''Element Types''
                 and command ''typs''
       GROUP   = element group number, see command ''grps''
       MATERIAL= element material number, see command ''mats''.
       NODE    = node number


Last Record (only FORMAT=0 | 1 (ascii), omitted for FORMAT=2):
Format:(1X,'-3')
Values: KEY
```

11.5 Parameter Header Record

To access the data use "prnt par". To manipulate data use "ds" option r.

```
Purpose: Stores informations related to datasets.
         ie. bondary conditions and loads
         They should consist of a keyword and a value

Format:(1X,'   1','P',A66)
Values: KEY, CODE, STRING
```

```
Where: KEY    = 1
       CODE   = P
       STRING = Keyword Value (ie: FORCE 1000.)
```

11.6 Nodal Results Block

To access the data use "ds". This command is also used to modify or create datasets. Values at nodes can be written with the "node" command.

```
Purpose: Stores values on node positions

1. Record:
Format:(1X,' 100','C',6A1,E12.5,I12,20A1,I2,I5,10A1,I2)
Values: KEY,CODE,SETNAME,VALUE,NUMNOD,TEXT,ICTYPE,NUMSTP,ANALYS,
        FORMAT
Where: KEY    = 100
       CODE   = C
       SETNAME= Name (not used)
       VALUE  = Could be frequency, time or any numerical value
       NUMNOD = Number of nodes in this nodal results block
       TEXT   = Any text
       ICTYPE = Analysis type
                0  static
                1  time step
                2  frequency
                3  load step
                4  user named
       NUMSTP = Step number
       ANALYS = Type of analysis (description)
       FORMAT = Format indicator
                0  short format
                1  long format
                2  binary format, values float
                3  binary format, values double

2. Record:
Format:(1X,I2,2X,8A1,2I5)
Values: KEY, NAME, NCOMPS, IRTYPE
Where: KEY    = -4
       NAME   = Dataset name to be used in the menu
       NCOMPS = Number of entities
       IRTYPE = 1  Nodal data, material independent
                2  Nodal data, material dependant
                3  Element data at nodes (not used)
```

```
3. Type of Record:
Format:(1X,I2,2X,8A1,5I5,8A1)
Values: KEY, NAME, MENU, ICTYPE, ICIND1, ICIND2, IEXIST, ICNAME
Where: KEY    = -5
       NAME   = Entity name to be used in the menu for this comp.
       MENU   = 1
       ICTYPE = Type of entity
                1  scalar
                2  vector with 3 components
                4  matrix
               12  vector with 3 amplitudes and 3 phase-angles in
                   degree
               14  tensor with 6 amplitudes and 6 phase-angles in
                   degree
       ICIND1 = sub-component index or row number
       ICIND2 = column number for ICTYPE=4
       IEXIST = 0  data are provided
                1  data are to be calculated by predefined
                   functions (not used)
                2  as 0 but flaged by cgx
       ICNAME = Name of the predefined calculation (not used)
                ALL  calculate the total displacement if ICTYPE=2
This record must be repeated for each entity.

4. Type of Record:  (not used)
This record will be necessary in combination with the request for
predefined calculations. This type of record is not allowed in
combination with binary coding of data.
Format:(1X,I2,2I5,20I3)
Values: KEY,IRECTY,NUMCPS,(LSTCPS(I),I=1,NUMCPS)
Where: KEY    = -6
       IRECTY = Record variant identification number
       NUMCPS = Number of components
       LSTCPS = For each variant component, the position of the
                corresponding component in attribute definition


5. Type of Record:
The following records are data-records and the format is repeated
for each node.

In case of material independent data

- ascii coding:
Following records (ascci, FORMAT=0 | 1):
 Short Format:(1X,I2,I5,6E12.5)
```

```
 Long Format:(1X,I2,I10,6E12.5)
 Values: KEY, NODE, XX..
 Where: KEY  = -1 if its the first line of data for a given node
              -2 if its a continuation line
       NODE  = node number or blank if KEY=-2
       XX..  = data

- binary coding:
 Following records (ascci, FORMAT=2):
 (int,NCOMPS*float)
 int and float are ansi-c data-types
 Following records (ascci, FORMAT=3):
 (int,NCOMPS*double)
 int, float and double are ansi-c data-types
 Values: NODE, XX..
 Where:
       NODE   = node number
       XX..   = data

In case of material dependant data
REMARK: Implemented only for NMATS=1
- first line:
Short Format:(1X,I2,4I5)
Long Format:(1X,I2,I10,3I5)
Values: KEY, NODENR, NMATS
Where: KEY    = -1
       NODENR = Node number
       NMATS  = Number of different materials at this node(unused)
- second and following lines:
Short Format:(1X,I2,I5,6E12.5)
Long Format:(1X,I2,I10,6E12.5)
Values: KEY, MAT, XX, YY, ZZ, XY, YZ, ZX ..
Where: KEY    = -2
       MAT    = material-property-number if KEY=-2 (unused)
       XX..   = data


Last Record (only FORMAT=0 | 1 (ascii), omitted for FORMAT=2):
Format:(1X,'-3')
Values: KEY
```

12 Pre-defined Calculations

Listing of the automatically calculated additional results.

12.1 Von Mises Equivalent Stress

Entity name: Mises

$$\sigma_{vM} = \frac{1}{\sqrt{2}}\sqrt{(\sigma_x - \sigma_y)^2 + (\sigma_y - \sigma_z)^2 + (\sigma_z - \sigma_x)^2 + 6\tau_{yz}^2 + 6\tau_{zx}^2 + 6\tau_{xy}^2}$$

12.2 Signed Von Mises Equivalent Stress

Entity name: sgMises
Uses the signum of the absolute greatest principal stress σ.

12.3 Von Mises Equivalent Strain

Entity name: Mises

$$\epsilon_{vM} = 2/3 * \frac{1}{\sqrt{2}}\sqrt{(\epsilon_x - \epsilon_y)^2 + (\epsilon_y - \epsilon_z)^2 + (\epsilon_z - \epsilon_x)^2 + 6\epsilon_{yz}^2 + 6\epsilon_{zx}^2 + 6\epsilon_{xy}^2}$$

12.4 Signed Von Mises Equivalent Strain

Entity name: sgMises
Uses the signum of the absolute greatest principal strain σ.

12.5 Principal Stresses

Entity names: PS1, PS2, PS3, worstPS

The principal stresses σ are named PS1, PS2, PS3. From the three principal stresses σ the absolute maximum value will be calculated and named worstPS. For example if a node has the three values 100, 0 and -110 MPa then -110 MPa would be shown. The three principal stresses σ_1 σ_2 σ_3 are derived from the following equation:

$$\begin{bmatrix} \sigma_{xx} - \lambda & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} - \lambda & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} - \lambda \end{bmatrix} \begin{bmatrix} nx \\ ny \\ nz \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

They are given by the three roots of the equation (stress tensor is symmetric: $\sigma_{xy} = \sigma_{yx}$ etc.):

$$\sigma^3 - (\sigma_{xx} + \sigma_{yy} + \sigma_{zz})\sigma^2 + (\sigma_{xx}\sigma_{yy} + \sigma_{yy}\sigma_{zz} + \sigma_{zz}\sigma_{xx} - \sigma_{xy}^2 - \sigma_{yz}^2 -$$

$$\sigma_{zx}^2)\sigma - (\sigma_{xx}\sigma_{yy}\sigma_{zz} + 2\sigma_{xy}\sigma_{yz}\sigma_{zx} - \sigma_{xx}\sigma_{yz}^2 - \sigma_{yy}\sigma_{zx}^2 - \sigma_{zz}\sigma_{xy}^2) = 0$$

12.6 Principal Strains

The same algorithm is applied to the strain tensor as outlined for the stress in "Principal Stresses".

12.7 maxShear Stresses

Entity name: maxShear
This is the maximum shear-stress:

$$maxShear = 0.5 * max((\sigma_1 - \sigma_3), (\sigma_1 - \sigma_2), (\sigma_2 - \sigma_3))$$

12.8 Contact Stresses

Entity name: Snorm
This is the stress component normal to the surface to which the node belongs.

12.9 Contact Strain

Entity name: Enorm
This is the strain component normal to the surface to which the node belongs.

12.10 Cylindrical Stresses

Entity names: SXX, STT, SRR, SXT, STR, SRX

The Cylindrical Stresses are truly cartesin stresses in a cylindrical system regarding the node-position relative to the axis of the cylindrical system. The stress-tensor is rotated individually for each node. They are calculated on demand (see trfm).

12.11 Weighted Error

Entity name: vMR
The element stress gradient based estimated error (requested with 'ERR' in ccx) is multiplied by the vMises ratio:

$$vMR = STR * vMisesStress(node)/vMisesStress(model)$$

13 Meshing rules

The mesher geneates at first nodes on points and then on lines between its end points. That means that a line which is supposed to be meshed with a 3 node beam element generates and references only 1 node! Then nodes in the interior of surfaces are generated. So when a surface is made of 4 lines with each of a division of 2 then the surface generates and references just one node! At last the bodies generate nodes in its interior. Again, if the body uses 6 surfaces as described before it generates and references only one node. If such a body is stored in a set and the user needs all nodes which are used by it, it is necessary to extend the set by the points, lines and surfaces which are used by the body. Their nodes are then also referenced by the set. This can be done with the

“comp” command. The same logic is used when creating the elements (surfaces generate shell elements and reference them, bodies generate volume elements).

If a tet mesh is required and the user meshes at first the surfaces and then generates the tet elements using the command “mesh 'setname' tet” only the 'setname' and the set 'all' reference the tet elements and the newly generated additional nodes! So an eventual target set has to be opened by the user before he starts the mesher. In case of unstructured meshes the mesh density of the inner area or volume is defined with "elty" (or "qmsh"). Also the locking of the surface or body is defined with that command (the element type of a locked entity can not be changed until unlocked). The applied mesh parameters are stored using the command "mshp". Usually the user does not use that command directly, it is just written by the program to store the information.

The way the mesher works has the consequence that only bodies who share a single common surface are connected at each node were they touch each other. Three surfaces with their points and lines are shown in figure 11. When sweeping this surfaces in 'y' direction three bodies will be generated. The line L006 generates a surface between the bodies at the position of A001 and A002. Both bodies will reference this surface and will therefore share its nodes. So all elements will be connected at that location. But a body using the surface A005 will use a surface generated by the combined line of L001 and L002 extruded in 'y' direction! This surface will not be used by the bodies generated by A001 and A002 since they will use the surfaces generated by the separately extruded lines L001 and L002. So bodies at the position of A001 and A005 will share the nodes of the common lines and points but not of the new surfaces at the location of line L001 (and not at the location of the new line at point D002). In the end at some inner locations gaps will exist.

Some rules must be fulfilled before a geometry is meshable (see mesh). For linear elements (ie. qu4 or he8), the sum of all divisions (see div) of each surface must be even. In case of quadratic elements (ie. qu8 or he20) this sum must be divisible by 4 without residue. Opposite edges of a given surface might have different divisions. For example on the left side of a given surface the division is 8 and on the right side it is only 4. But only two opposite surfaces of a body can use this feature. These surfaces are called top and bottom surfaces. All other surfaces of this body must have unique divisions on opposite edges. In case of 3 sided surfaces it is necessary to apply a minimum division sufficient for two elements along the edge. The only exception is the element tr3u (see elty) which allows a division of one.

A body can not be meshed when the shape of the body is very far from being brick-like. The body might be subdivided to improve the shapes of the single ones. There is a restriction for the definition of five- or seven-sided bodies. The first two surfaces in the body-topology (see gbod) have to be defined in the same order. That means the first line of the first surface has to be connected with the first line in the second surface by one of the remaining surfaces. This is always the case if the body is a product of a "swep" command.

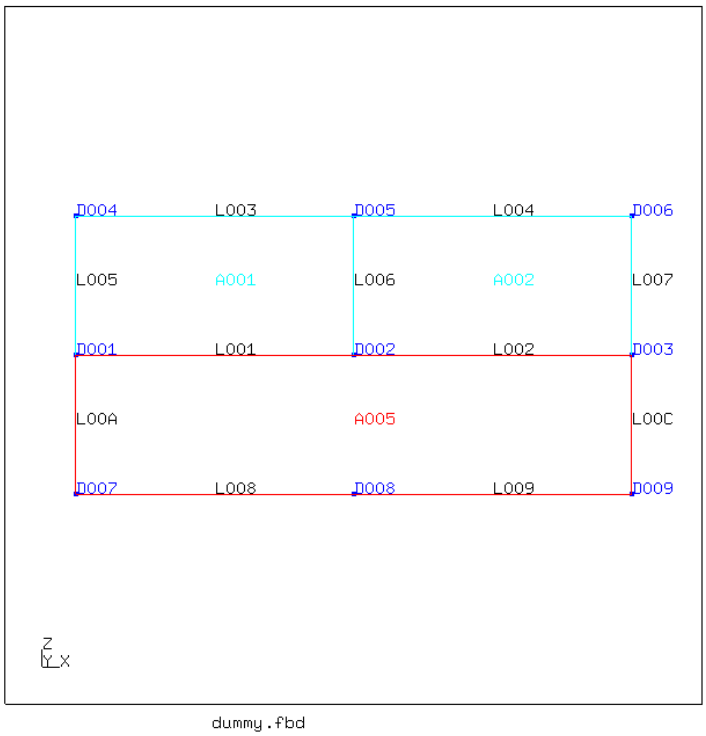


Figure 11: unconnected bodies

14 User-Functions

The user might define his own functions to manipulate the mesh or the results with the user function stored in file "userFunction.c". See the command "call" how to call a user function. The user can generate new nodes, elements or datasets or extend or manipulate existing datasets or interfaces to other software. The file "userFunction.c" includes an example which calculates the hydrostatic stress and stores the result in a new dataset. In case the user has no access to a compiler he may solve his task by using the in build command language. The section "User File Parser" can be used as an example.

15 Internal Setnames

The program generates internal sets for various purposes. For example to store boundary coditions which were found in an input file or to store elements which fail a certain criterion etc.

```
/* special sets, generated by cgx  */
```

```
/* internal sets should have a leading "+" or "-" */
/* Setnames with leading "-" are not shown with qenq */
/* Setnames with leading ":" are not shown with prnt */
```

- "-BLR" /* surfaces to be rendered by the structured mesher */
- "-BNUR" /* save all bad nurbs */
- "-BSUR" /* save all bad surfs */
- "+CF" /* contact faces */
- "-COPY" /* */
- "-DEP" /* used for proj */
- "-DMPC" /* dependent nodes of equations */
- "+EDGE" /* all model edge nodes */
- ":ETMP" /* save all temporary elements, deleted by next call to adjustDrawNodes(1) */
- "-GLUR" /* unmeshable surfaces to be rendered by the glu routine */
- "-HIGHL" /* highlights identified entities */
- "-IMPC" /* independent nodes of equations */
- "-IND" /* used for proj */
- "-MESH" /* */
- "-NJBY" /* bad elements */
- "-NOEL" /* dependent nodes for which no matching indep-elem was found but it was conn to a close one */
- "-NOMESH" /* failed surfaces during 2d meshing */
- "-NOMPC" /* not used dependent nodes */
- "-NOPRJ" /* failed projection */
- ":NOSAVE" /* entities not saved when set all is saved */
- "-ORI" /* */
- "+PENTR" /* stores nodes of one mesh which penetrate a second mesh */
- "+PLT2D" /* selected nodes for the menu-graph (2d-plot) */
- "+THRS" /* nodes with a value above or below the threshold */

- "-TMP" /* temporary buffer for several purposes */
- "-UORI" /* save all unoriented items */
- "-WARN" /* problematic surfaces during 2d meshing */
- "-ZAP" /* entities to be deleted */

A Known Problems

A.1 Program does not show the geometry after startup

If no 'plot' or 'plus' commands are included in the command files (fbd or fbl files) then nothing might be visible after startup. Pleas use the plot command 'plot l all' and 'frame'.

A.2 Program is not responding

If the program seems to hang then leave the window with the mouse pointer and go in again. If that does not help then probably a command is waiting for input. Stay with the mouse pointer inside the window and press the "q" key several times. Another very popular error is to move the mouse-pointer into the konsole were the typed commands and the protocol is visible. Please, the mouse-pointer MUST stay in the main-window during typing! The user might use the menu fuction "Toggle CommandLine" or the command "view cl" to switch the command line from the konsole to the graphic's window.

A.3 Program generates a segmentation fault

Write a mail to the author and, if possible, add the input-file.[3].

B Tips and Hints

The following collection will give you background information and procedures to deal with common situations.

B.1 How to change the format of the movie file

Use a shell command like convert to split up movie.gif with multiple layers into multiple frames, as seperate jpeg files:

```
convert movie.gif %d.jpg
```

Convert any series of multiple jpeg files into WMV format with:

```
mencoder mf://*.jpg -mf w=800:h=600:fps=25:type=jpg -ovc lavc -lavcopts
 vcodec=wmv2 -o movie.wmv
```

To convert a video file into avi format use this command:

```
mencoder <videofile> -ovc lavc -lavcopts vcodec=flv -of avi -o movie.avi
```

In general:To convert a video file from one format into the other, use the program "mencoder". It comes with the mplayer packages.

```
mplayer movie.wmv
```

B.2 How to get the sets from a geo- or ccx-inp file for post-processing

Quite often it is useful to have the already defined sets from the pre-processing or/and the calculation available when doing the post-processing of results. The sets defined in the ccx-input file (.inp) can either be read together with the results (.frd) at start-up:

cgx results.frd input.inp

or the user may read them during run-time

read input.inp nom

(see "read"). When it comes to the sets defined in the geo file a slightly different approach is needed. The geo file must be read before the result file. So the user starts cgx in the build mode

cgx -b geofile.fbd

meshes the model (see "mesh") and then read the results

read results.frd nom

and optionally the input file. Be aware that the parameter "nom" is essential here. Use "prnt" to get an overview over the available sets.

B.3 How to define a set of entities

Some knowledge is necessary to efficiently select entities (nodes, points ..). As a golden rule in complicated situations never try to create a set just by "adding" the entities to the set (see qadd). It is much better to catch a bigger group which includes certainly the wanted ones. Then display the set with the plot command and remove all unwanted entities with the qrem command. Entities are selected if at least one pixel of it is inside the selection rectangle. But there is one exception: Only the lower left corner of any text (names of points etc.) can be selected. Surfaces can also be selected by picking its shaded interiour (see "rep)". You might add all lower entities (as points for example) by typing

"comp set do". This is necessary for example if you had selected surfaces and you want to move them in space. Only points have locations and therefore nothing will happen unless you completed the set by the related points with "comp" command.

There is another type of set called sequence. The data-structure is the same but with one exception, the data keep their order in which they were selected. This type of set is used for splines (see "qlin" and "line") or in combination with the "graph" command if values along a path should be displayed. Use "prnt sq" to list all existing sequences and use "qseq" or "seqa" to create them. Up to now they are only used to store nodes and points. As sets they will be written to a file if fbd format is specified. In this case also node-sequences can be stored.

B.4 How to enquire node numbers and values at certain locations

A very common problem is how to get the exact value on a node position during post-processing. To actually get the value add the element edges to the view (see Toggle Element Edges) and type qenq and press the RETURN key (during typing the mouse-pointer MUST stay in the main window, do NOT move the pointer into the konsole). Then move the mouse-pointer over the location of a node and press the "n" key. The node-number, the value at that node and the location will be written in the konsole from which the cgx was started. See also the "enq" command for batch controlled value extraction.

B.5 How to select only nodes on the surface

Some times you need to select nodes only on the surface of the mesh. This can be done when the mesh is displayed in the surface mode (see "Toggle Surfaces/Volumes") using the menu-entry "Show Elements With Light". To find node positions more easily add the element edges to the view (see Toggle Element Edges).

A different way uses the element-faces (see "plot" with parameter "f"). Store the desired faces in a new set (qadd) and complete the set downwards ("comp set do"). Check the selected nodes with "plot n setname". Or you add all faces at once in a new set with "seta surfs f all" before completing the set downwards.

B.6 How to write values to a file

When you need to write certain values (results) to a file you may use either the "send" command which allows to write results in certain formats or a command file which uses the 'echo' system command in combination with the "valu" command. The following list shows the general structure of such a command file:

- Choose the relevant dataset (menu or ds command)

- Add the relevant node to a set ('qadd setname' or 'seta setname n number')
- Open the stack: stack on
- Place node-values on the stack: prnt se setname
- Store the node: valu nod1 pop
- Store the value: valu val1 pop
- Write to file: sys echo Node nod1 Value val1 >| file.txt
- Or optionally: sys printf (consult the man page for the format) >| file.txt

See also "if" and "while" and see the demo in "If and while demo" on how to work with this commands. They are useful for automatic processing.

B.7 How to generate a user dataset

It is possible to calculate and store new results with the cgx command language. The new dataset and its entities is created with the command "ds". The data are written with the "node" command to the new dataset. The "valu" command is used for calculations and data handling. The example "Data storage in a user dataset" determines the normal direction on all nodes on the free surface of the mesh and stores this information in a new dataset. See also the sections "How to write values to a file", "How to process results".

B.8 How to generate a time-history plot

So called time history plots can be created based on a previous displayed sequence of data-sets (see "Toggle Dataset Sequence") with the command "graph". It is also possible to use only the command line. For example:

- ds 1 e 3
- ds 1 2 10
- graph set t

will produce a time history plot for the nodes stored in set over all loadcases from 1 to 10 for the entity nr 3. Or

- graph set t DISP D1

will display the displacement in direction 1 for all loadcases. For more details and other options look into the "graph" command description.

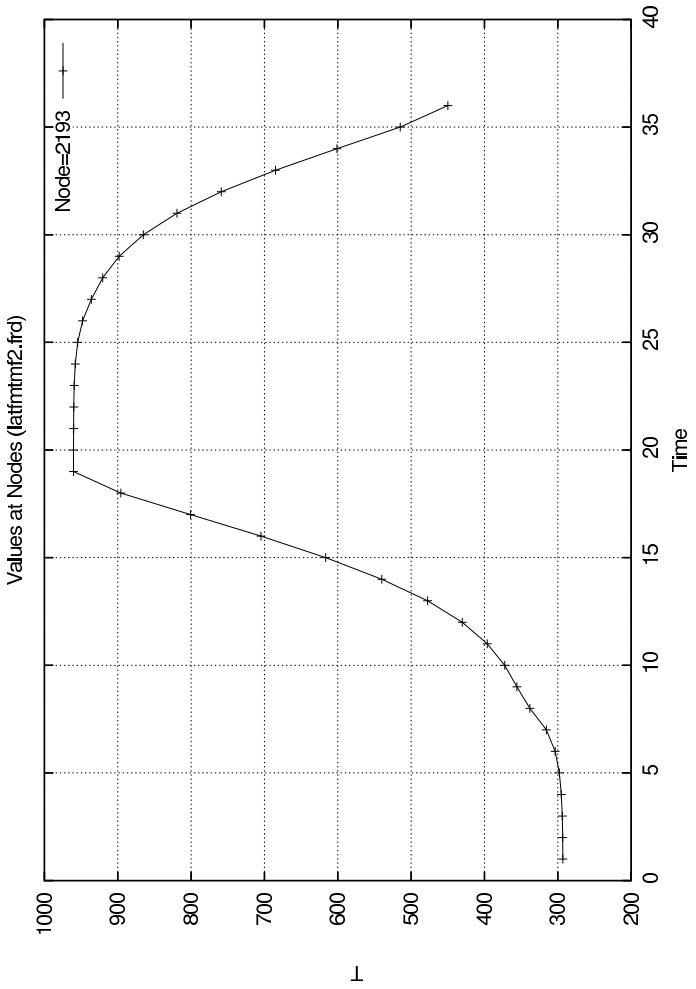


Figure 12: example of a time-history plot

B.9 How the mesh is related to the geometry

Very often you need the embedded nodes, element-faces or elements of geometric entities to apply boundary conditions. If you understand the underlying concept you will be able to get them more easily.

Geometric entities are the mothers of nodes, faces and elements (so to say) and will remember them. In turn if an entity is not the mother of a given mesh-entity it will not remember it. Therefore a body will only know about nodes which are not placed on surfaces, lines or points. A surface will only know about nodes which are not placed on lines or points. And so on.

Therefore if you need the nodes on a surface and not only the ones just inside

the surface, then create a set with this surface and do a "comp set do". This will add the lines and points together with their nodes to the set.

On the other hand if you have only a node and you need the geometric entity in which the node is embedded you might also type "comp set do".

B.10 How to change the order of elements

Use the command "mids" to change from second order to first order or vice versa. In this case the amount of elements will not change. Or use the command "send" with the parameter "quadlin" to change from second order to first order. But in this case each second order element will be splitted in 8 first order elements.

B.11 How to connect independent meshes

Sometimes it is advisable to "glue" independent meshes together instead of trying to create one big seamless-mesh. Or you just want to create a contact formulation were you know that no separation will happen. In this situations you might use equations (also called mpc's) which connect one dependent node with one or more independent nodes. The independent side should be coarser than the dependent side to avoid gaps in the connection. See the command "send in combination with the option "areampc" how to create such connections. Sliding and fixed contact as well as press-fits can be modelled. The necessary sets should be defined based on geometry not on the mesh. In this case the definition will be stored with the "save" or "exit" command and can be used after the next program call. If no sets for the connections are at hand you may use the "neigh" command to generate them.

Background: The movement of each location in an element (or on its surface) is fully described by the movement of its associated nodes and its shape-function. In an iterative algorithm element-coordinates are varied until the real-world coordinates of a dependent node are matched. Based on the element coordinates the shape function gives the participation (weight) factors of the independent nodes (the coefficients in the mpc's). This approach delivers valid results as long as the dependent node is not located outside of the independent element. Therefore the location of the dependent node has to be modified in such cases. The dependent will be moved to the surface of the independent element (again decribed by its shape-function) and the procedure is repeated.

B.12 How to define loads and constraints

Loads and constraints are not saved in any database. They are just created and written to a file with the "send" command. But the sets which are needed for the definition are stored together with the geometry if you type the "save" command. Of course the sets must have been defined based on geometry and not based on mesh entities like nodes because mesh-entities are not saved with the "save" command. You must know that geometry-sets know also their mesh entities after a "mesh" command. If you store your commands to write the

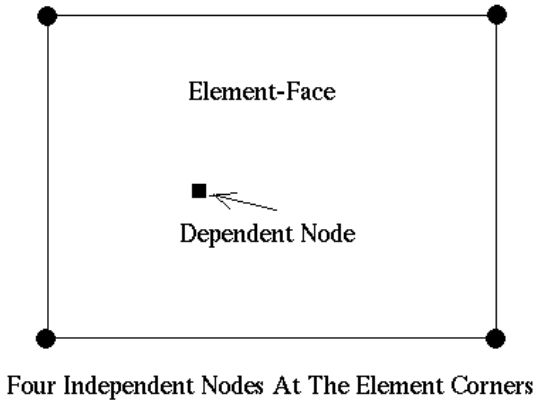


Figure 13: Dependent node on element face to create a connection

boundary conditions in a file you can easily repeat them by just reading this command-file (see "read"). Several types of loads can be applied. That is forces, pressure, thermal-heat-coefficients and radiation etc. For unsupported loads write a set of element-, node- or face-labels (parameter "names") and apply the desired load to that set in the solver input-file. Other boundary conditions like single-point-constraints for structural, thermal and fluid-calculations can be written. The sliding condition (parameter "slide") were nodes are bound to the element-faces to which they belong can be used for structural calculations or for fluid calculations. In the later case it forces the flow to follow the surface of the elements. This is necessary in inviscid calculations to prevent the fluid to penetrate the walls. Results from a previous calculation can be written in the solver input format for further calculations (parameter "ds", "tmf" etc.). For example temperatures for thermal mechanic fatigue analysis or velocities, pressures and thermals for a restart of a cfd calculation.

B.13 How to map loads

Values at nodes can be mapped (interpolated) from a second mesh with the "map" command (2D and 3D). This second mesh (data source or "master") is usually available as a result-file from a previous calculation. The "read" command with the "add" parameter can be used to include this file in the

current model. Solver-input files (inp) and result files (frd) can be used as a data source. The command will add an offset to the nodes and elements so that existing nodes and elements will not be overwritten. The original mesh is the target or "slave" of the mapping process. The mapping process will add the interpolated (mapped) values to the dataset(s) which originated from the master mesh. The mapped values can be visualized by selecting the mapped dataset and entity and then by plotting the slave elements or faces with "plot fv slave". An example of the necessary commands for the mapping process (surfaces to surfaces) is listed below:

```
REMARK: Be aware that you have to create sets of nodes and
        faces for surface mapping! They will be called 'master'
        and 'slave'

Open the target mesh with cgx.
Define a set with the slave nodes or faces to which values
should be mapped (for example pressure):
  qadd slave t10 (here with a tolerance of 10 deg)
  comp slave do (to extend the set by the referenced faces or
                 nodes)

Then open a set and read the model with the values (pressure):
  seto source
  read result.frd add
  setc

If necessary move 'source' in space to match the position of the
'slave' (see ``move'').

Add the faces to the 'source' set
(now only nodes and elements are already stored in 'source'):
  comp source do
  plot f source

Define the master set:
  qadd master t10  (catch the right faces showing the values)
  comp master do

Map the values of dataset 1 with:
  map slave master surf ds1

Check the mapping with:
  ds 1 e 1
  plot fv slave

Write the file with the mapped values:
```

```
  send slave abq pres ds1 e1

Check the file ‘‘slave_ds1e1.dlo’’ with an editor and use it
in an inp-file.
```

If the master values are not available in either inp or frd format but in any other format which can be read by cgx (isaac, openFoam ..) you may write them in frd format with the "send" command to be usable as a master.

B.14 How to run cgx in batch mode

Most commands can be executed in batch-mode. Actually if you read a file with geometry (fbd-file) then you run cgx in batch mode already! You just have to add "exit" or "quit" at the end of the file and it will be obvious. You might use the "read" command in such a command-file to reference other command-files or to read files with mesh-entities for whatever purposes. In such a way you can create and modify geometry or meshes in batch-mode or evaluate results in batch-mode. But you need a graphic-capable computer because cgx needs it even if no graphic output is requested. Nevertheless the pop up of the graphic window can be suppressed by starting cgx with the "-bg" parameter. In this mode the performance is higher since no graphic related actions are performed. Even during foreground operation the user has the possibillity to execute commands in the background mode when he switches this mode on with "asgn bg on". Especially when reading commands from a file it increases the execution time significantly. After reading it has to be switched of with "asgn bg off".

Some commands only make sense in a batch file like "if" and "while". Please have a look into "If and while demo" on how to work with this commands. Results can be extracted and stored to a file, see "How to write values to a file".

Be aware of the 'examples' directory. It holds examples on how to use the command language as a programming language. The 'ifwhiledemo' generates points in a loop and shows how to write cgx generated values to a file. The 'userDataset' shows how to generate a user dataset for a temperature field based on the z coordinate of the nodes.

Regard the "stop" command to stop file execution and the "cont" command to re-start reading. This is very helpfull during creation of such a batch file.

It should be noted that a successive 'save' or 'exit' command will overwrite the batch file if it has the file-extension 'fbd'. To prevent this the user should use a different extension like 'fbl' instead of 'fbd'.

When executing commands which increase the used space then it might happen that geometry is clipped and can not be seen or accessed anymore. In such a case a "frame" command is needed. This command is usually automatically triggered. Especially in batch mode without graphic output it solves sometimes strange situations.

When the command "init" or "wsize" is used then they have to be used in the very first line. Otherwhise the window dimensions take only effect after the batch file was completely parsed and executed.

B.15 How to process results

The node results can be modified or used to generate derived results. There are basically three levels of operations.

The easiest one is to just multiply with a factor, add a number or use an exponent to all node results of a certain dataset with the "ds" command.

The next level is using variables which can be used to do calculations based on results and can be written back to an existing dataset or a new one. This requires a good knowledge of the way cgx works and of the necessary commands ("if", "while", "stack", "valu", "enq", "prnt", "ds", "sys" etc.). In this way data can be written to a file with the "sys" command (depending on the operation system using echo or printf) were an external program might be invoked by the "sys" command which in turn could provide data for cgx.

Finally, complex operations can be coded in C in the file userFunction.c. The user function is activated by the "call" command. A new compilation of cgx is required unless a dynamically linked library is used.

B.16 How to deal with CAD-geometry

In general hexahedra-elements perform better than tets but if the mesh should be derived from a cad-geometry it is often more convenient to create a tetrahedra mesh as to modify or rebuild the geometry to make it meshable with hexahedra-elements. The following section gives some hints which alternatives are available:

- The CAD format is supported by a stand alone translator (see the calculix home pages). In this case the user can mesh the surfaces with structured- [14] (elty setname tr3 etc.) or unstructured triangles [15] (elty setname tr3u etc.). The user might modify this surface mesh until he is happy with it. The surface mesh can then be filled with tets created by an external auto-mesher called from within cgx (mesh setname tet). The cgx uses the tet-mesher from NETGEN [4] or TETGEN [5] for this task. The command "asgn" is used to switch between them.

  Therefore cgx can only generate a tet-mesh if one of these programs are accessible (TETGEN like glut is included in the cgx package). For the full functionality replace the original ng_vol source file in the NETGEN package with the modified program ng_vol from the cgx-distribution and build it again. This version regards a target element size. Tetgen is able to regard that size unchanged.

- For CAD models in STEP- or IGES-format you may consider to use a tet-mesher like NETGEN [4] which often generates quite nice tet-meshes with very few user interaction. You can read this meshes with cgx and combine them with cgx-geometry and meshes. Then create your boundary conditions etc. You might read the native-netgen format (.vol) instead of abaqus-format because this includes the 2D meshing regions in separate sets which can be used to apply boundary conditions ("cgx -ng file.vol", see "Program Parameters"). See "prnt" how to list the available sets.

- A simple step-reader is integrated in cgx (cgx -step filename). It can deal with points and lines. This is sufficient for axis-symmetric structures like a rotor but the experienced user might also use the following approach for more complex geometries: You start with a file containing a 2D-section, subdivide it in mesh-able surfaces and sweep it in the 3rd dimension to create your geometry (see swep). If features exist in the 3rd dimension then this features must be included in the 2D-section. Sweep the 2D-section to the location were the feature starts, then right to the end and at last to the end of the geometry. Delete the unwanted bodies before and after the feature. You might project the sweped sections to a target surface if the feature is shaped in the 3rd dimension (see proj).

  Sometimes the geometry consists of several (maybe even identical) parts which are arranged on different positions in space. This is called assembly. When cgx is started with the parameter "-stepsplit" instead of "-step" it will write the single parts to separate directories using their original coordinate systems. The user can prepare meshable geometry or meshes based on them. But he must use the filenames which he finds there. The final meshed assembly can be build afterwards by calling the fbl-file which was written by cgx. This fbl-file contains the original step-commands to position and eventually duplicate the single geometries/meshes from the subdirectories.

The following section describes the process to generate a tet-mesh with cgx based on a cad model:

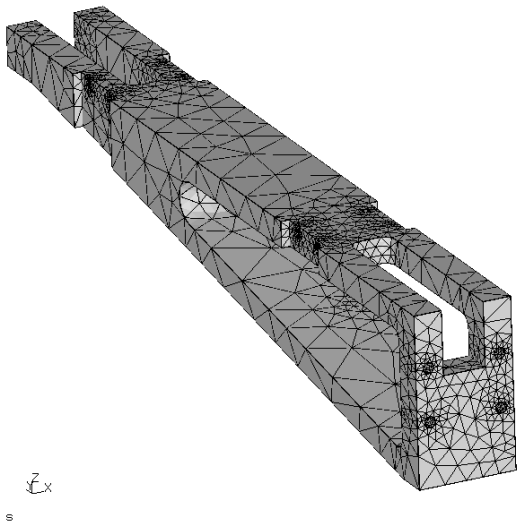


Figure 14: CAD-geometry meshed with tet-elements

- Install and use the propper interface program to convert the cad format to fbd format (You find example data (vda,iges,step) in CalculiX/cgx_x.x/examples/cad)

  ```
  (ie: vda2fbd halter.vda > halter.fbd).
  ```

  - Remark regarding vda2fbd: In some cases the header of the vda file causes trouble and must be modified to make the program run.
  - Remark regarding cad2fbd: The cgx can not deal with surfaces which use the same line on two sides (a cylinder made of just one surface). Please use the parameter -sa with cad2fbd. It splits such surfaces in two (cad2fbd -sa cadfile). If problems occur call cad2fbd without arguments and play along with the listed parameters.

- Start cgx with that file (cgx -a halter.fbd). Usually some warnings appear on the screen but the program will fix that automatically (triggered by the parameter -a). After all geometry has been read the program will merge points and lines to close the volume (body). WARNING: If no body exists the user has to manually merge all line end-points individually for all parts. Otherwhise intentionally unconnected parts could be merged at some locations. If the model consists of only one part a 'merg p all' should work if no geometry changes are planned. To be on the save side the user should prefer to use the following commands to merge only line endpoints:

  ```
  seta b1 l <set>
  comp b1 e
  merg p b1
  ```

  Finally the line divisions are determined based on default values. Use "plot lc all r 3" to see the lines in red 3 digit whide with node spacing as defined by the line divisions. The current default values lead to approximatelly 200 elements over the length of the model. The default values are defined in cgx.h, see GTOL_NODE_DIST and GTOL. You may use now the command div in the 'auto' mode to change the element sizes to your needs individually in each relevant region.

- If problems occur the user might start the program with option -b instead of -a (cgx -b halter.fbd). Then no automatic pre-processing is done and the user has to prepare the geometry manually. This commands are executed when starting with -a:

  ```
  # merge only points which are referenced by lines
  seta LBUF l all
  comp LBUF e
  merg p LBUF
  # delete zero length lines
  del l0 all
  # set the line divisions
  div all auto
  ```

The user may use this command sequence when he reads a cad fbd file during run time ("read cadfile.fbd"). See the commands merg, div, qdiv, qlin, comp and rep.

- A default element type was assigned to all surfaces (tr6u) if you had started cgx with parameter -a. You may change the element type with elty. In most cases it is advisable to use this triangles first (tr6u) and not directly tetraeders (te10) because a surface mesh can be interactively improved before the tets are generated. Remark: You can assign tets only to a set holding one or more bodies. The body related surfaces get a preliminary triangle element type assigned as a basis for the tets. If no body exists you may create one with "body ! all". You assign the tet element type with the "elty" command (elty setname te10).

- All surfaces should be shaded and filled with triangles. This has to be done manually by typing "mesh all" (see mesh) or "rep all" which just calculates the interior shape of the surfaces. In most cases "mesh" is preferable since it does both at the same time. Since the surface meshing of a CAD geometry can be very time consuming a default number of threads is used when the model is opened in the auto mode (cgx -a file). The actual number of threads will be listed when using the command "prnt info". The user might change this value with the command asgn before he starts meshing. The default is set in cgx.h, see NTHREADS.

- Use then "plus si all" to display the shaded surfaces. If a surface points inwards it is not illuminated and appears only dark grey. Flip the surfaces in a way that its normal direction points outwards. Use "qflp" abd press the "a" key to swith to the "auto" mode and then select a correctly oriented surface by pressing key "s", all other surfaces will immediatelly use the same orientation (if it's a closed volume). Add now the element faces with "plus f all".

- If some surfaces could not be meshed then modify the divisions of the lines of this surfaces (see "qmsh", key 'l') and/or chose the structured element type tr6 for this surfaces (see "qmsh", key 'x'). Or change the element-size (see "qmsh", key 'h' or 't') inside the surfaces. The command "qmsh" fixes parts of the surface-mesh in a convenient way since this command combines several other ones. For convenient usage of "qmsh" display the surfaces "plot si all" together with faces "plus f all" and add the lines with "plus lc all r 3").

  Remark: In some cases surfaces are not meshable when you had combined two surfaces before with "qmsh". The referenced NURBS might be too small to cover the whole new surface. Without such a related nurbs no unstructured mesh can be created. If structured elements are no solution you may delete the related NURBS from the surface definition with "qsur" using key 'b'. Then store the surface in a set and create a new NURBS with "nurs ! setname". The surfaces stored in the set will produce NURBS

based on the Coons algorithm [14]. If the surface shape matches a primitive shape you may also use the "qshp" or "shpe" command to generate such a shape and assign it to the surface.

- Check the mesh. If the command "prnt info" lists "ed:0" then no model edges exist and the mesh is closed and ready to be used for tet meshing. If edges exist then the surface mesh is not closed which means that not all shells have exactly one neighbor at each edge. This free edges exist usually at the location of lines and the problem can often be solved by changing the divsion of the underlying lines. The edges are displayed by default but might be hidden by other entities. To increase the size of the representation up to 10 pixels use "view ed 10". This can be seen on figure 15. Additionally the nodes at this free edges are stored in the set "+EDGE". Then chose the line under one edge with "qmsh" and change the line division to a higher or lower number until the edge disapears. Actually cgx detects this edges after meshing and tries to solve the problem by increasing the division of the related lines incrementally, but only a few times. The remaining lines (and all nodes at this lines, not just the nodes on free edges(opposite to "+EDGES")) are stored in set "-EDGE" and can be displayed as well for manual adjustments.

- Generate the tet mesh ("mesh all tet"). A target-size for the tet-elements can be specified if "ng_vol" from the cgx distribution was installed (for example "mesh all tet 2.5" will define a maximum element-size of "2.5"). If the model consists of several unconnected parts separate them in single sets and mesh one after the other (see seta, qadd, qrem, comp). Check the mesh and correct bad elements:

```
prnt eq all
mids -BJBY force
prnt eq all
(repeat that procedere until no elements are stored in -BJBY any more)
```

- Create sets of nodes, faces or elements to create boundary conditions (if you had not created them already based on the geometry) and export them with "send".

- Create the input deck for ccx with an editor and start the calculation.

- Look at the results with cgx (cgx results.frd).

Additional remarks:

- It should be mentioned that only the set which was used for tet meshing will hold the created tet elements. The user has to use "seto" if other target sets are needed.

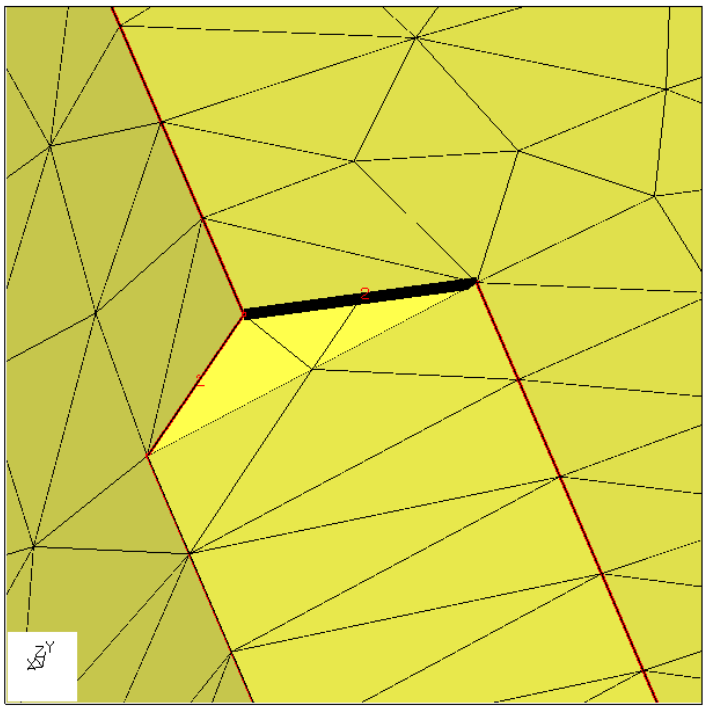


Figure 15: Visible (black) edge over lines and faces. Such edges exists if the surface mesh is not closed and not meshable with tets. Here two elements meet just one element at the black edge.

- In cases were certain regions of the model need different inner mesh densities the user might use the "split" command to subdivide the model in disjunct bodies which can be meshed with independent density. Skilled users might use bodies which share common surfaces. This way 'equations' or 'contact' can be avoided.
- If a te10 meshed body shares surfaces with a he20 meshed body equations connecting tet related nodes which are not used by the hex elements are generated and will be written into the mesh file together with the nodes and elements. But this feature works only if all related bodies are meshed together with just one 'mesh' command. So create and use a set holding all adjacent bodies for use with the 'mesh' command. This feature requires in any case a body for the tet meshed structure. Bodies which do not share common surfaces with the tet body can be meshed in separate steps.
- For cyclic symmetric boundary conditions it is preferable to have the same

mesh on both connected surfaces. To archive this the user should not mesh one of two sides. Instead he copies the elements of the meshed side to the location of the un-meshed side (see "copy"). Since the element orientation of the target side has to be inverted it is necessary first to change the orientation of the source side (see "flip" option 's' or "qflp"). Then copy the elements to the target side and merge the nodes in that region (usually just "merg n all"). Then the orientation of the source side has to be inverted again (see "flip" option 's' or "qflp")). Now the tet mesh can be generated.

- Surfaces will not be deleted when the user deletes lines which define complete holes in existing surfaces. But it is necessary that all lines of a certain hole are deleted at once. If only a partial hole is deleted, the surface is deleted as well. It is proposed to use zap to delete the affected lines.

- Sometimes a surface is not meshable because a line runs right into it (a sliver, see figure 16). The following commands can be used to remove it:

```
qlin (select the lines with 'a', 'x')
qpnt (select the line endpoint 'p', in the figure marked with
      DL2M and place it over the other endpoint 'p')
      The line(s) are now deleted. Often such a surface is only
      meshable with a regular mesh (tr6), so change the type
      and mesh then:
qmsh (select the surface 's' and change the type 'x' and mesh
      'm')
```

B.17 How to check an input file for ccx

A quick check of a ccx input-file can be done with cgx by reading the file with the option -c (cgx -c file.inp). After startup all defined node-, element-, and surface-sets are availabe together with internal sets which group togeter certain entities according to their purposes. The following internal sets will be created if appropriate data were found:

- +bou(DOF): Created if *BOUNDARY is found. All affected nodes are stored in +bou were +bou(DOF) store just the nodes which are constrained in the related DOF number.

- +dep(BOU): Created if *EQUATION is found. All dependent nodes are stored in +dep.

- +ind(BOU): Analog to +dep.

- +clo[nr]: Created if *CLOAD is found. The nr counts the number of appearance.

- +dlo[nr]: Created if *DLOAD is found.

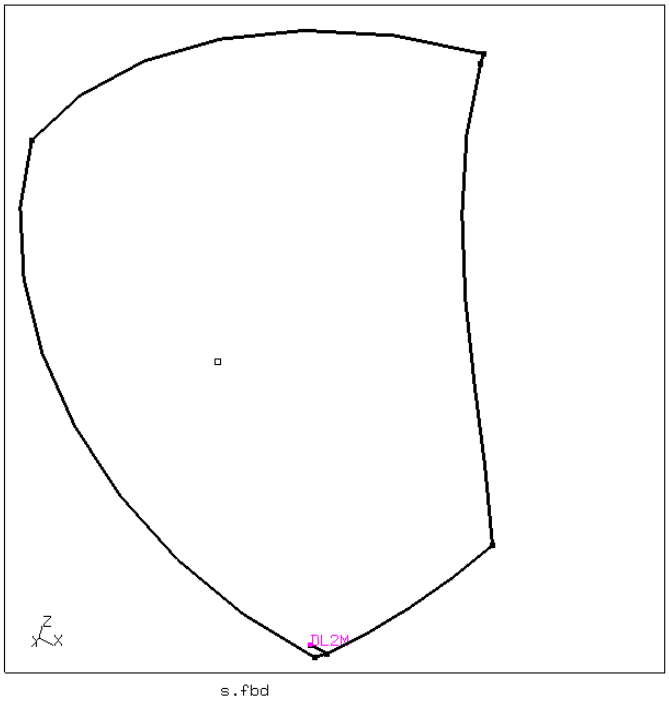


Figure 16: CAD-surface with a sliver or scratch

- +mpc[nr]: Created if *MPC is found.
- +rad[nr]: Created if *MPC is found.
- +flm[nr]: Created if *FILM is found.
- +cflx[nr]: Created if *CFLUX is found.
- +dflx[nr]: Created if *DFLUX is found.
- +tie[1—2]: Created if *TIE is found. The set +tie1 stores the slave entities and +tie2 the master. The single sets which define the contact areas are linked together so if identified with the “qenq” command the referenced (linked) opposing set is listed in the konsole.
- +trans...: Created if *TRANSFORM is found. The name consists of the definition of the transformation.
- +[elty]: Created for all known element types like *C3D20.

REMARK: Internal setnames start either with a ’-’ or ’+’ (see “Internal Setnames”). The names which start with a ’-’ are not listed with “qenq”.

Check the element quality with “Show Bad Elements” or use “eqal” to set thresholds and “plot” or “prnt” to actually plot or list the affected elements.

Forces regard their referenced coordinate-system (*TRANSFORM). The values are automatically transformed into the global cartesian system so that the vectors point in the correct direction.

Multiple load definitions inside one *STEP on a single entity will sum up (ccx compatible). This applies to cflux, dflux, cload, dload.

B.18 Remarks Concerning Ansys

The cgx is capable to write the following mesh entities to files (see “send”, the necessary key-parameters are listed in brackets below):

- Nodes and Elements
- Sets of nodes and elements (nam)
- Single point constraints (spc)
- Equations (areampc)
- Pressure (pres)
- Temperatures (ds, all datasets with just one entity will be written as temperatures)

The resulting files have to be combined with the help of an editor and extended by material-data and the necessary controll-commands.

So far results can not be read.

B.19 Remarks Concerning Code Aster

From Paul CARRICO (2005/02/12)

Brief presentation of CODE ASTER: Code Aster is an implicit solver under GPL licence from the French company EDF (Électricité de France).

Code Aster and its documentations is downloadable at the following address:

http://www.code-aster.org

(NOTA : the documentation is in French at the moment but many users are translating it in English and in German)

The following list is not exhaustive but it briefly presents the capabilities of the solver:

- Mechanical modeling: Code aster allows linear and non linear calculations in static, dynamic, acoustic etc. Many mechanical laws are implemented in the solver such as damage, fatigue, creep, viscosities (elastic, plastic) etc. For isotropic and non-isotropic materials (orthotropic ones for example). Because of EDF fields, the materials used in Code Aster are essentially Metallic ones and Geo material one, but there are probably some others.

- Thermal and thermomechanical calculations: The Thermal solver performs linear and non-linear calculations for pure thermal but also for thermomechanical simulations.
- Input interface: EFICAS is the input interface for coding the input file, but it's not a pre-processing as you can find in many commercial code.
- Tools: Many tools are provided with Code Aster: HOMARD for mesh refinement, GIBI and GMSH for post-processing.

CGX to ASTER export format (i.e HOWTO use this export format) The export format allows to export meshes and sets from CGX to Code Aster for linear and quadratic 1D, 2D and 3D elements. For this, just type: 'send all aster' for exporting a complete mesh. The file will have the '.mail' extension. Type 'send set aster nam' for exporting the GROUP-NO (node group) and the GROUP-MA (mesh group) which compose the set. The file will have '.nam' extension

The later item is particularly useful to apply boundary conditions onto (DOF, pressure, force, displacement, temperature, etc.). Since of these boundary conditions are coded with EFICAS in the input file, I thought it was not necessary to develop another features than the 2 previous ones !

So far results can not be read.

B.20 Remarks Concerning dolfyn

Some support for dolfyn (a free cfd code) was provided by Runar Tenfjord. In the CalculiX/cgx_(nr)/examples/dolfyn directory is a patch for the dolfyn source-code included. This patch enables dolfyn to write frd-result files which can be visualized with cgx. There is also an example which allows cgx to be used as an pre-processor for dolfyn. The mesh can be written in dolfyn format with the command "send".

B.21 Remarks Concerning Duns and Isaac

If you intent to create a 2D-mesh for the cfd-code duns or isaac you have to watch out that all surfaces are created in the same order. That means that all surfaces must be defined clockwise if you look in z direction. The block-structure can be detected only in this case. You can check the mesh by simply mesh it with linear shell-elements and display them. All elements must be illuminated if looking against z. A later "qflp" or "flip" command will not cure wrong oriented surfaces since it only changes the "sign" in their definition and not the basic edge sequence, which is necessary here. The results of a calculation can be opened by specifying the parameter (-duns2d -duns3d -duns2dl -duns3dl -isaac2d -isaac3d) and the filename without any extention(cgx -isaac2d RAE2822). See also "Program Parameters", "mesh", "send" and the airfoil-example in the distribution. Please read the comments for duns and isaac in the "send" command description.

Be aware that duns and isaac use block meshes which must be created using the set 'all'. So the use of set 'all' together with the parameter 'block' is mandatory when the block structure is needed! Nevertheless the user might only assign an element type to a sub set so that only a part of the geometry will be meshed (see "mesh").

If the solver-format "duns" is used then related numbers of surfaces (3D) or lines (2D) will be written to file "duns.bou". This information is necessary to apply boundary-conditions to duns. The numbers are used in the connectivity file "duns.conn" which will be created together with the mesh. The user has to refer this numbers in the "duns.script2" file when assigning boundary conditions.

When using the cgx parameter 'periodic' with the send command it is necessary to deactivate the function 'CHKCUT' in the isaac main.F file:

```
modifications Wittig 13. Jul 21
main.F 933  // deactivate the coord check to enable periodic boundaries
c         CALL CHKCUT (IBLK1, IDIM(IBLK1,ILVL), JDIM(IBLK1,ILVL),
c     1                KDIM(IBLK1,ILVL), R(IR1),
c     2                IBLK2, IDIM(IBLK2,ILVL), JDIM(IBLK2,ILVL),
c     3                KDIM(IBLK2,ILVL), R(IR2), ICUTS(1,ICUT,ILVL),
c     4                CUTNAM(ICUT), IERRCD)
```

Otherwhise the performed check would stop the program.

B.22 Remarks Concerning ls-dyna

The cgx is capable to write the following mesh entities to files (see "send", the necessary key-parameters are listed in brackets below):

- Nodes and Elements
- The *PART command to link a setname with an element property ID
- Node- (*SET_NODE_LIST) and element sets (*SET_SOLID)
- Surfaces (*SET_SEGMENT)

The resulting files have to be combined with the help of an editor and extended by material-data and the necessary controll-commands.

```
# a mesh with sets (here COMP1,COMP2,SPC,SURF)
# has to be available
grpa 1 COMP1 dyn
grpa 2 COMP2 dyn
send all dyn
seta SPC#1 n SPC
seta SURF#2 n SURF
send SPC#1 dyn nam
send SURF#2 dyn sur
# assemble the input file:
```

```
sys echo "*KEYWORD" >| input.dyn
sys cat COMP1_group1.dyn >> input.dyn
sys cat COMP2_group2.dyn >> input.dyn
sys cat all.dyn >> input.dyn
sys cat SPC#1.nam >> input.dyn
sys cat SURF2.sur >> input.dyn
sys echo "*END" >> input.dyn
```

B.23 Remarks Concerning Nastran

The cgx is capable to write the following mesh entities to files (see "send", the necessary key-parameters are listed in brackets below):

- Nodes and Elements
- node displacement coordinate system (see command "csysa")
- Single point constraints (spc)
- Equations or RBEs (areampc, to glue components. A previous command "asgn" defines if mpcs or rbes will be created)
- RBE2 (mpc, for rbe-spiders)
- Pressure (pres, so far only CHEXA8)
- Temperatures (ds, all datasets with just one entity will be written as temperatures)

The resulting files have to be combined with the help of an editor and extended by material-data and the necessary controll-commands.

The f06-file with results can be read (so far only CHEXA with displacements and stresses).

B.24 Remarks Concerning NETGEN

It is not necessary to write the mesh in abaqus format if you use netgen as a mesher. The native netgen format (.vol) can be read by cgx (cgx -ng file.vol) as well. The netgen mesh format (.vol) includes the surface-patches which were defined by the edges of the model and used for the generation of the volume-mesh. This patches can be used to define boundary conditions or loads. The nodes and faces of this patches are stored in sets named "+set[nr]". To get an overview over the patches type "prnt se". To see were the patches are located type "plot f all" and use the "PAGE_DOWN"-key to scan through all sets.

A netgen surface mesh can be written based on faces of elements. The faces of hex, tet, quad and tria elements are triangulated and written in the stl format which can be read by using the netgen-gui or the stand-alone netgen mesher format (file.ng). This mesher can be found in the netgen sub-directory nglib and

is named ng_vol. It will create tet4 elements which use and keep the shape of the provided tri3 elements. To improve the meshing results with the netgen-gui the user could create own edges based on the stl triangles or read and manipulate the netgen created edges with cgx and then write them back. To read the edges: In NETGEN open the stl-doctor and go in the edges menu. There delete all edges with "all undefined" then load the edges with "load edgedata" and activate them with "candidate to confirm".

B.25 Remarks Concerning OpenFOAM

The mesh can be written in OpenFOAM polyMesh format with the command "send". If you work in the polyMesh-directory of the OpenFOAM case then all mesh-related files will be already in place. So far the physical-type is not written in the boundary file as it is not mandatory. The results of an OpenFOAM calculation can be viewed by specifying the parameter -foam and the case (the relative or absolute path including the directory-name of the case). See also "Program Parameters", "mesh".

B.26 Remarks Concerning Samcef

From Paul CARRICO (2006/04/17)

BASIC TUTORIAL FOR HOWTO USE THE SAMCEF EXPORT FORMAT

```
1- Definition of the points
K: pnt p0 0 0 0
K: pnt p1 1 0 0
K: pnt p2 0 1 0
K: pnt p3 2 1 0
K: plot pa all

2- Definition of the lines
K: plus l all
K: qlin (link the points p0 p1 p3 p2 p2 p0)

3- Creation of the first surface
K: qsur

4- Creation of l0 (between p0 &p1) and l1 (between p1 &p3) sets
K : qadd lo
K : qadd l1

4- Creation of the 2 other surfaces
K : swep l0 l1 tra 0 -3 0
K : swep l1 l1b tra -3 0 0
```

```
5- Creation of the SYMETRY set
K : plot s all
K : qadd SYMETRY (use both a and rr keys to select all the
                  surfaces)

6- Creation of the volumes
K : swep SYMETRY s1 swep tra 0 0 1 (all the volume will be
    automatically created)

7- Looking for common points, lines and surfaces
In the order :
K : merg p all
K : merg l all
K : merg s all

8- Creation of the LOAD set and ANCHORAG one
K : qadd LOAD (use rr keys to select the surface)
K : qadd ANCHORAG (use rr keys to select the surface)

NOTA : It's easy to verify the different sets ; for example :
K : plot b all (you can see all the volumes)
K : plus s LOAD (you can see the set LOAD)
K : plus s ANCHORAG
K : plus s SYMETRY

9 - Mesh
K : plot ld all
K : div all mult 2
K : elty all HE20 (to specify HEXAHEDRA with 20 nodes)
K : elty LOAD qu8 (to mesh the set LOAD otherwise no quads will
                   be created)
K : elty ANCHORAG qu8
K : elty SYMETRY
K : mesh all (to mesh the part with all.dat name)
K : send all sam (to export the mesh into Samcef format)
K : send LOAD sam nam (to export groups into Samcef format)
K : send ANCHORAG sam nam (see previous remark)
K : send SYMETRY sam nam

10- Modifications
It's possible now to make some modifications :
a- open all.dat file with your favorite text editor (Vi for me)
b- open ANCHORAG.nam & the SYMETRIC.nam files and do the same as
   previously
c- concatenate under Linux the files using the following schema :
        cat all.dat LOAD.nam > s1.m
```

```
        cat s1.m ANCHORAG.nam > s2.m
        cat s2.m SYMETRY.nam > part.dat
(all the sx.m files will be erased afterward)
d- open PART.dat file and go to the end => then add RETURN
e- the mesh file now works with Samcef


Another interesting way : add for each .nam file an input in your
 bank file:
input ''part.dat''
input ''LOAD.nam''
input ''ANCHORAG.nam''
etc. ...

11 IMPORTANT REMARK

After, it's possible to modify the mesh into BACON (extrusions,
etc. ...);
that's why the element hypothesis is not added at the end of the
file ;
=> you must define the element definition AFTER the last mesh
 modification (.HYP MINDLIN)

12- Comments

if you've any remark or any comment or any suggestion to improve
this export format, please send a mail to paul.carrico_at_free.fr
```

So far results can not be read.

C Simple Examples

The following listings show simple geometry input-files. The pictures show this geometry together with their labels and the generated mesh. The models were made based on three points. Two points defined one axis of rotation and one was the basis of several "swep" and "merg" operations. In case of the sphere the surfaces on the pole had to be redefined using only three lines per surface.

C.1 Disc

```
 PNT py      -0.00000        1.00000        0.00000
 PNT p0      -0.00000       -0.00000        0.00000
 PNT P001     0.70711       -0.00000       -0.70711
 PNT P003    -0.00000       -0.00000       -1.00000
 PNT P005    -0.70711       -0.00000       -0.70711
```

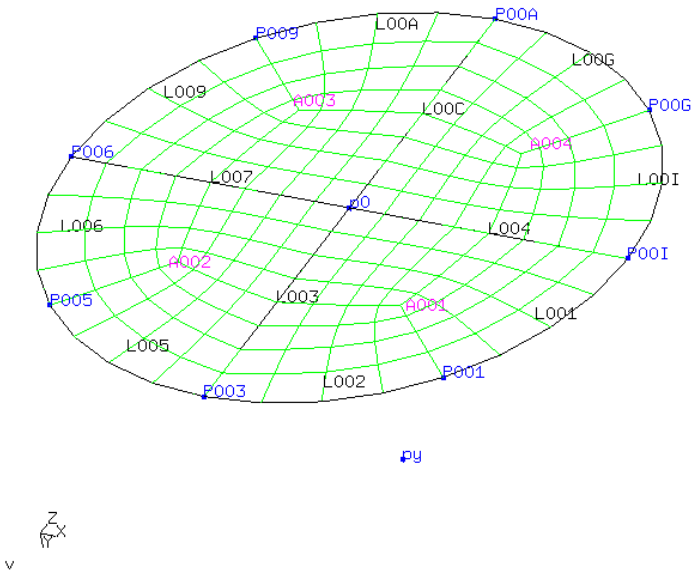


Figure 17: disc made of four 90 degree segments

```
PNT P006    -1.00000       -0.00000        0.00000
PNT P009    -0.70711       -0.00000        0.70711
PNT P00A     0.00000       -0.00000        1.00000
PNT P00G     0.70711       -0.00000        0.70711
PNT P00I     1.00000       -0.00000       -0.00000
LINE L001 P00I P001 p0 4
LINE L002 P001 P003 p0 4
LINE L003 P003 p0 8
LINE L004 p0 P00I 8
LINE L005 P003 P005 p0 4
LINE L006 P005 P006 p0 4
LINE L007 P006 p0 8
LINE L009 P006 P009 p0 4
LINE L00A P009 P00A p0 4
LINE L00C P00A p0 8
LINE L00G P00A P00G p0 4
LINE L00I P00G P00I p0 4
```

```
GSUR A001 + BLEND  - L003 - L002 - L001 - L004
GSUR A002 + BLEND  - L007 - L006 - L005 + L003
GSUR A003 + BLEND  - L00C - L00A - L009 + L007
GSUR A004 + BLEND  + L004 - L00I - L00G + L00C
ELTY all QU4
```

C.2 Cylinder

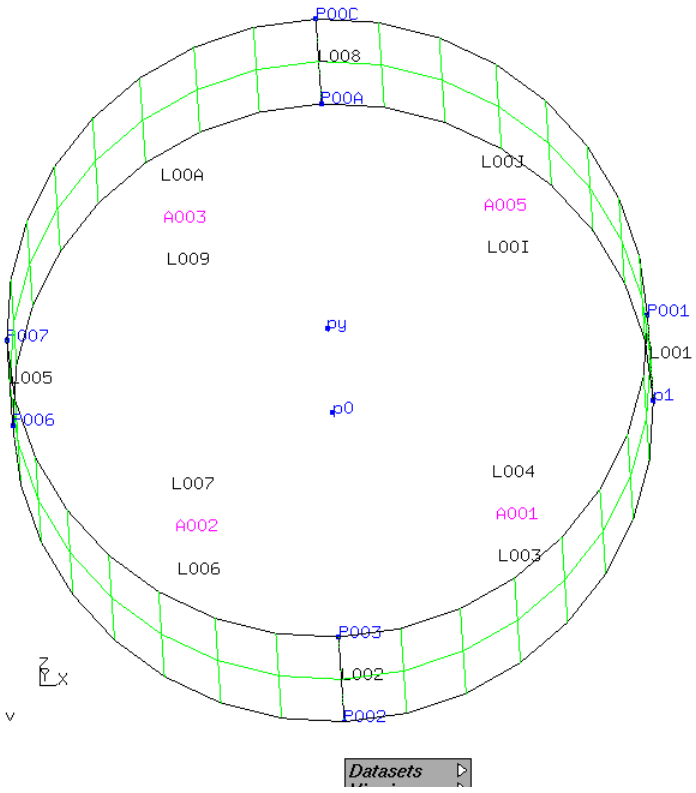


Figure 18: cylinder made of four 90 degree segments

```
PNT p0      -0.00000       -0.00000        0.00000
PNT py      -0.00000        1.00000        0.00000
PNT p1       1.00000       -0.00000        0.00000
PNT P001     1.00000        1.00000        0.00000
PNT P002    -0.00000       -0.00000       -1.00000
PNT P003    -0.00000        1.00000       -1.00000
PNT P006    -1.00000       -0.00000        0.00000
PNT P007    -1.00000        1.00000        0.00000
PNT P00A     0.00000       -0.00000        1.00000
```

```
PNT P00C     0.00000        1.00000        1.00000
LINE L001 p1 P001 2
LINE L002 P002 P003 2
LINE L003 p1 P002 p0 8
LINE L004 P001 P003 py 8
LINE L005 P006 P007 2
LINE L006 P002 P006 p0 8
LINE L007 P003 P007 py 8
LINE L008 P00A P00C 2
LINE L009 P006 P00A p0 8
LINE L00A P007 P00C py 8
LINE L00I P00A p1 p0 8
LINE L00J P00C P001 py 8
SHPE CYL1 cyl p0 py 1.
GSUR A001 + CYL1  - L001 + L003 + L002 - L004
GSUR A002 + CYL1  - L002 + L006 + L005 - L007
GSUR A003 + CYL1  - L005 + L009 + L008 - L00A
GSUR A004 + CYL1  - L008 + L00I + L001 - L00J
ELTY all QU4
```

C.3 Sphere

```
PNT py      -0.00000        1.00000       -0.00000
PNT p1       1.00000       -0.00000       -0.00000
PNT P001     0.70711       -0.00000       -0.70711
PNT P003    -0.00000       -0.00000       -1.00000
PNT P006     0.70711        0.50000       -0.50000
PNT P008    -0.00000        0.70711       -0.70711
PNT P00C     0.70711       -0.00000       -0.00000
PNT P00K     0.70711        0.70711       -0.00000
PNT P00L    -0.00000       -0.00000       -0.00000
PNT P00N    -0.00000        1.00000       -0.00000
LINE L001 p1 P001 P00L 8
LINE L002 P001 P003 P00L 8
LINE L003 p1 P006 P00L 8
LINE L004 P006 P008 P00L 8
LINE L006 P001 P006 P00C 8
LINE L008 P003 P008 P00L 8
LINE L00A p1 P00K P00L 8
LINE L00C P00K P00N P00L 8
LINE L00G P006 P00K P00C 8
LINE L00J P008 P00N P00L 8
SHPE SPH1 sph P00L 1.
GSUR A005 + SPH1  - L003 + L001 + L006
GSUR A002 + SPH1  - L002 + L006 + L004 - L008
GSUR A006 + SPH1  + L003 + L00G - L00A
```

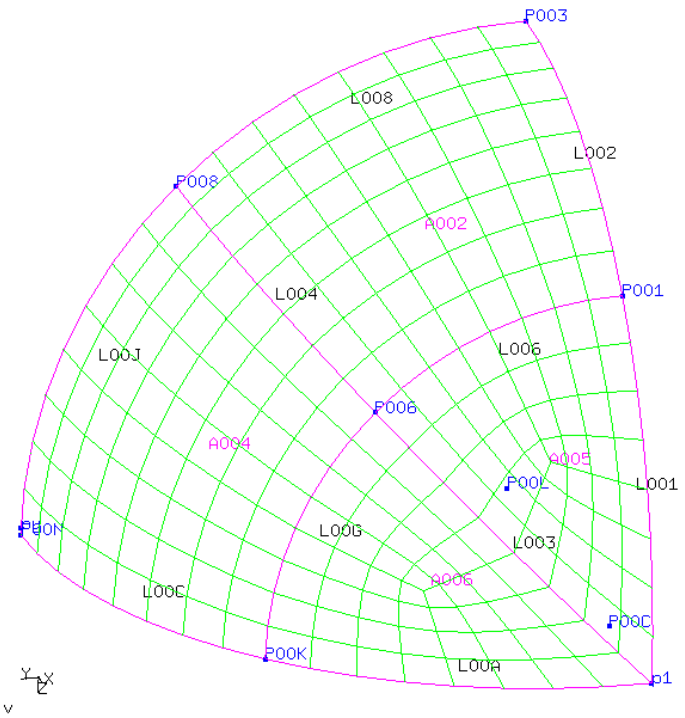


Figure 19: Segment of a Sphere

```
GSUR A004 + SPH1  - L004 + L00G + L00C - L00J
ELTY all QU4
```

C.4 Sphere (Volume)

```
PNT py       0.00000        1.00000        0.00000
PNT p1       1.00000        0.00000        0.00000
PNT P006     0.70711        0.50000       -0.50000
PNT P008     0.00000        0.70711       -0.70711
PNT P00C     0.70711        0.00000        0.00000
PNT P00K     0.70711        0.70711        0.00000
PNT P00L     0.00000        0.00000        0.00000
PNT P00N     0.00000        1.00000        0.00000
LINE L001 p1 P00L 8
LINE L002 P00L P008 8
LINE L003 p1 P006 P00L 8
LINE L004 P006 P008 P00L 8
```

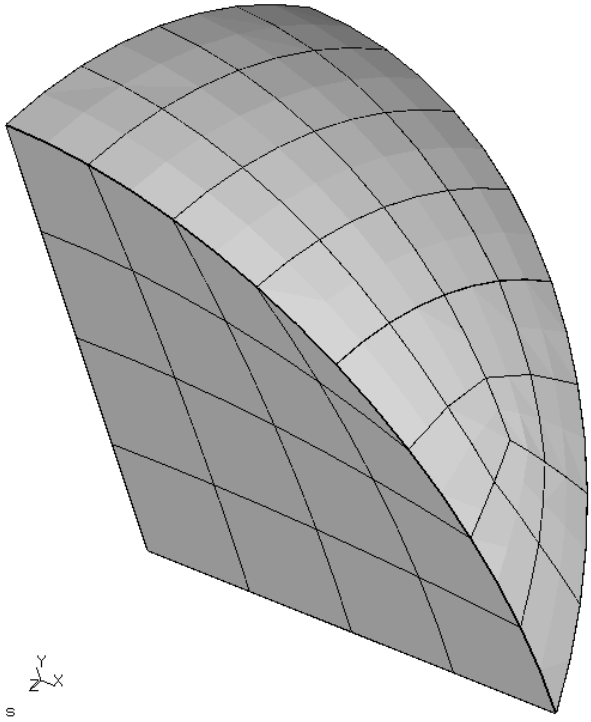


Figure 20: Segment of a Sphere (Volume)

```
LINE L005 P00L P00N 8
LINE L00A p1 P00K P00L 8
LINE L00C P00K P00N P00L 8
LINE L00G P006 P00K P00C 8
LINE L00J P008 P00N P00L 8
SHPE SPH1 sph P00L 1.
GSUR A001 + BLEND - L003 + L001 + L002 - L004
GSUR A002 + BLEND - L005 - L001 + L00A + L00C
GSUR A006 + SPH1 + L003 + L00G - L00A
GSUR A004 + SPH1 - L004 + L00G + L00C - L00J
GSUR A003 + BLEND + L002 + L00J - L005
GBOD B001 NORM + A006 - A003 - A004 + A002 + A001
ELTY all HE20
```

C.5 Sphere (Thin Volume)

```
# written by dichtstoff
```

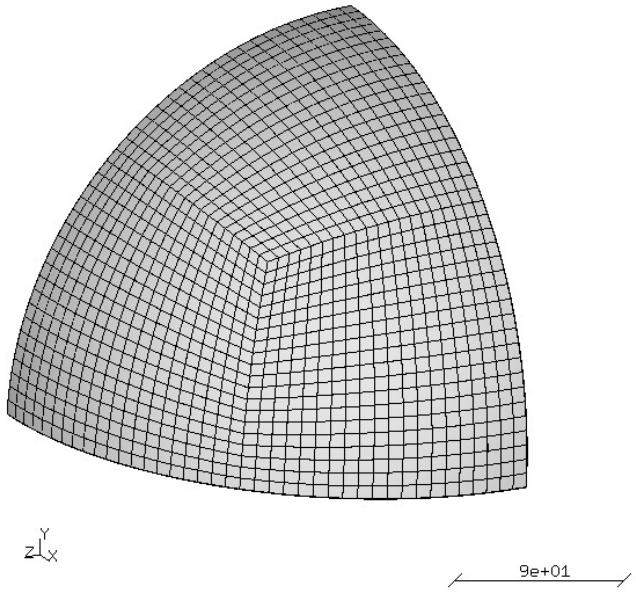


Figure 21: Segment of a Sphere (Thin Volume)

```
# Spherical shell made of 3D elements
# thickness of sphere
# valu sphth
# radius of the sphere
# valu sphrd
# valu points of sphere
# valu sphv1
# valu sphv2
# amount of element with line division
# valu lidiv
#######################################
valu sphth 1.93
valu sphrd 200
valu sphth        * sphth  +0.5
valu sphthoffseto + sphrd sphth
valu sphthoffseti - sphrd sphth
valu sphrdn       * sphrd  -1.0
```

```
valu lidiv 20

PNT base     0.00000       0.00000       0.000000000
PNT D001     0.00000       0.00000       sphthoffseto
PNT D002     0.00000       0.00000       sphthoffseti
PNT D003     0.00000       sphthoffseto  0.000000000
PNT D004     0.00000       sphthoffseti  0.000000000
PNT D005     sphthoffseto  0.00000       0.000000000
PNT D006     sphthoffseti  0.00000       0.000000000
PNT roty     0.00000       sphrdn        0.000000000
PNT rotx      sphrdn       0.00000       0.000000000

LINE ! D001 D002 1
LINE ! D003 D004 1
LINE ! D005 D006 1

seta L001 l L001
seta L002 l L002
seta L003 l L003

copy L001 L004 rot y 45
copy L002 L005 rot x 45
copy L003 L006 rot z 45

seta roty p roty
seta rotx p rotx

move roty rot z  45
move rotx rot z -45

copy L006 L007 rot roty rotx -35.264389683

LINE ! D001 D007 base lidiv
LINE ! D001 D009 base lidiv
LINE ! D002 D008 base lidiv
LINE ! D002 D00A base lidiv
LINE ! D003 D009 base lidiv
LINE ! D003 D00B base lidiv
LINE ! D004 D00A base lidiv
LINE ! D004 D00C base lidiv
LINE ! D005 D00B base lidiv
LINE ! D005 D007 base lidiv
LINE ! D006 D00C base lidiv
LINE ! D006 D008 base lidiv
LINE ! D007 D00D base lidiv
LINE ! D008 D00E base lidiv
```

```
LINE ! D009 D00D base lidiv
LINE ! D00A D00E base lidiv
LINE ! D00B D00D base lidiv
LINE ! D00C D00E base lidiv

SHPE SPHi sph base sphthoffseti
SHPE SPHo sph base sphthoffseto

GSUR A001 + BLEND - L006 + L00O + L007 - L00P
GSUR A002 + BLEND - L005 + L00M + L007 - L00N
GSUR A003 + BLEND - L004 + L00K + L007 - L00L
GSUR A004 + BLEND - L003 + L00G + L006 - L00I
GSUR A005 + BLEND - L003 + L00H + L004 - L00J
GSUR A006 + BLEND - L002 + L00C + L005 - L00E
GSUR A007 + BLEND - L002 + L00D + L006 - L00F
GSUR A008 + BLEND - L001 + L008 + L004 - L00A
GSUR A009 + BLEND - L001 + L009 + L005 - L00B
GSUR A010 + SPHo - L00G + L00H + L00K - L00O
GSUR A011 + SPHi - L00I + L00J + L00L - L00P
GSUR A012 + SPHi - L00E + L00F + L00P - L00N
GSUR A013 + SPHo - L00C + L00D + L00O - L00M
GSUR A014 + SPHi - L00A + L00B + L00N - L00L
GSUR A015 + SPHo - L008 + L009 + L00M - L00K
GBOD B001 NORM + A001 - A005 - A003 - A011 + A004 + A010
GBOD B002 NORM - A001 + A006 + A002 + A013 - A007 - A012
GBOD B003 NORM - A002 + A008 + A003 + A015 - A009 - A014

div all mult 2
elty all he20r
mesh all
plot f all n
```

C.6 Airfoil for cfd codes

All surfaces must be oriented in the same way. The sets are used to define areas for the boundary conditions.

```
PNT P002    -0.24688        0.00667        0.00000
PNT P003    -0.24375        0.00903        0.00000
PNT P004    -0.23750        0.01228        0.00000
PNT P005    -0.23125        0.01450        0.00000
PNT P006    -0.22500        0.01608        0.00000
PNT P007    -0.21250        0.01798        0.00000
PNT P008    -0.20000        0.01875        0.00000
PNT P009    -0.18750        0.01900        0.00000
PNT P00A    -0.17500        0.01888        0.00000
```

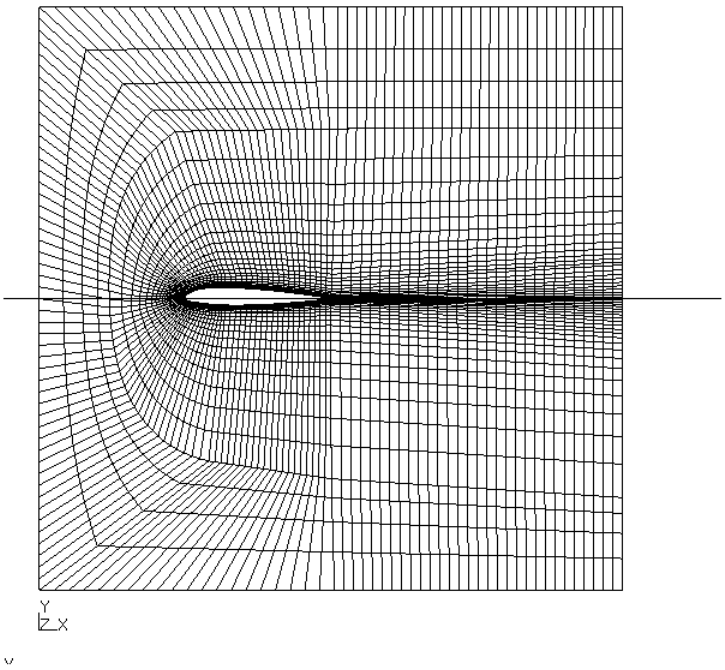


Figure 22: Airfoil for cfd codes

```
PNT P00C    -0.15000        0.01785        0.00000
PNT P00E    -0.12500        0.01602        0.00000
PNT P00G    -0.10000        0.01368        0.00000
PNT P00I    -0.07500        0.01090        0.00000
PNT P00J    -0.05000        0.00770        0.00000
PNT P00K    -0.02500        0.00420        0.00000
PNT P00L    -0.01250        0.00230        0.00000
PNT P00O    -0.25000        0.00000        0.00000
PNT P00P    -0.24688       -0.00308        0.00000
PNT P00R    -0.24375       -0.00427        0.00000
PNT P00S    -0.23750       -0.00565        0.00000
PNT P00T    -0.23125       -0.00653        0.00000
PNT P00V    -0.22500       -0.00730        0.00000
PNT P00W    -0.21250       -0.00875        0.00000
PNT P00X    -0.20000       -0.00993        0.00000
PNT P00Z    -0.18750       -0.01070        0.00000
PNT P010    -0.17500       -0.01115        0.00000
```

```
PNT P011    -0.15000       -0.01120        0.00000
PNT P012    -0.12500       -0.01043        0.00000
PNT P013    -0.10000       -0.00917        0.00000
PNT P014    -0.07500       -0.00750        0.00000
PNT P015    -0.05000       -0.00540        0.00000
PNT P016    -0.02500       -0.00308        0.00000
PNT P017    -0.01250       -0.00175        0.00000
PNT P019     0.00000        0.00000        0.00000
PNT P1      -0.50000       -0.50000        0.00000
PNT P2       0.50000       -0.50000        0.00000
PNT p3       0.50000        0.50000        0.00000
PNT p4      -0.50000        0.50000        0.00000
PNT P01A    -0.18162        0.01898        0.00000
PNT P01B    -0.18180       -0.01094        0.00000
PNT P046    -0.27025        0.01256        0.00000
PNT P059    -0.26599        0.28688        0.00000
PNT P049    -0.25144        0.02439        0.00000
PNT P05A    -0.35589        0.17566        0.00000
PNT P04C    -0.22636        0.03241        0.00000
PNT P04D    -0.20128        0.03643        0.00000
PNT P05C    -0.38027        0.00958        0.00000
PNT P04G    -0.12604        0.03833        0.00000
PNT P04H    -0.07588        0.03616        0.00000
PNT P04I    -0.02572        0.03231        0.00000
PNT P05D    -0.31932       -0.21136        0.00000
PNT P04L    -0.27652       -0.00154        0.00000
PNT P04M    -0.27025       -0.00803        0.00000
PNT P05F    -0.20962       -0.27840        0.00000
PNT P04P    -0.25373       -0.01567        0.00000
PNT P04R    -0.22676       -0.02143        0.00000
PNT P04T    -0.20124       -0.02394        0.00000
PNT P05G     0.01132        0.29145        0.00000
PNT P04W    -0.12604       -0.02508        0.00000
PNT P04X    -0.07588       -0.02519        0.00000
PNT P04Z    -0.02572       -0.02355        0.00000
PNT P05H     0.00065       -0.30887        0.00000
PNT P052    -0.18074        0.03754        0.00000
PNT P054    -0.18133       -0.02465        0.00000
PNT P056     0.00118        0.02891        0.00000
PNT P058     0.00010       -0.02250        0.00000
PNT P05I     0.01244        0.50000        0.00000
PNT P05J     0.00610       -0.50000        0.00000
PNT P05L     0.50000       -0.34112        0.00000
PNT P05M     0.50000        0.29206        0.00000
PNT P05N     0.50000        0.05780        0.00000
PNT P05S     0.50000       -0.05314        0.00000
```

```
PNT P05V     0.50000       -0.00217        0.00000
PNT P00N    -0.23448        0.01345        0.00000
PNT P02M    -0.23471       -0.00608        0.00000
PNT P03B    -0.24164        0.02804        0.00000
PNT P03C    -0.23405       -0.02029        0.00000
PNT P03E    -0.24536        0.00794        0.00000
PNT P03P    -0.24464       -0.00400        0.00000
PNT pl1     -1.00000        0.00000        0.00000
PNT pl2      1.00000        0.00000        0.00000
SEQA S006  pnt  P01A P00A P00C P00E P00G P00I P00J P00K P00L P019
SEQA S007  pnt  P019 P017 P016 P015 P014 P013 P012 P011 P010 P01B
SEQA S00W  pnt  P03B P049 P046 P04L P04M P04P P03C
SEQA S00R  pnt  P052 P04G P04H P04I P056
SEQA S00S  pnt  P058 P04Z P04X P04W P054
SEQA S00T  pnt  P05F P05D P05C P05A P059
SEQA S001  pnt  P01A P009 P008 P007 P006 P005 P00N
SEQA S00E  pnt  P03E P002 P00O P00P P03P
SEQA S00L  pnt  P052 P04D P04C P03B
SEQA S00A  pnt  P02M P00T P00V P00W P00X P00Z P01B
SEQA S00X  pnt  P03C P04R P04T P054
SEQA S002  pnt  P00N P004 P003 P03E
SEQA S00P  pnt  P03P P00R P00S P02M
LINE L003 P01A P052 910
LINE L00C P01A P019 S006 120
LINE L00E P019 P01B S007 120
LINE L004 p4 P1 150
LINE L05F P2 P05L -204
LINE L05S P05L P05S -912
LINE L05C P1 P05J 120
LINE L006 P01B P054 910
LINE L007 P019 P056 910
LINE L008 P058 P019 -910
LINE L00G P05S P05V -210
LINE L00N P03B P03C S00W 130
LINE L03R P052 P056 S00R 120
LINE L00I P05V P05N 210
LINE L03S P058 P054 S00S 120
LINE L04V p4 P059 -204
LINE L04W P059 P052 -912
LINE L04X P054 P05F 912
LINE L04Z P05F P1 204
LINE L050 P05F P059 S00T 150
LINE L052 P059 P05G 120
LINE L054 P05F P05H 120
LINE L056 P05H P058 -912
LINE L058 P056 P05G 912
```

```
LINE L059 p3 P05I 130
LINE L05A P05I p4 120
LINE L05D P05J P2 130
LINE L05V P056 P05N 130
LINE L05I P05M p3 204
LINE L05L P05N P05M 912
LINE L05Z P058 P05S 130
LINE L06C P019 P05V 130
LINE L06F P05M P05G 130
LINE L06H P05G P05I 204
LINE L06I P05L P05H 130
LINE L06J P05H P05J 204
LINE L001 P01A P00N S001 -210
LINE L00A P03E P03P S00E 120
LINE L00K P052 P03B S00L 110
LINE L009 P02M P01B S00A 210
LINE L00O P03C P054 S00X 110
LINE L002 P00N P03E S002 -205
LINE L00L P03P P02M S00P 205
LINE cl pl1 pl2 120
LINE L005 P00O P019 120
LCMB C001  + L001 + L002 + L00A + L00L + L009
LCMB C004  + L00K + L00N + L00O
GSUR A001 + BLEND + L003 + C004 - L006 - C001
GSUR A002 + BLEND + L006 - L03S + L008 + L00E
GSUR A003 + BLEND + L00C + L007 - L03R - L003
GSUR A004 + BLEND - L008 + L05Z + L00G - L06C
GSUR A005 + BLEND + L06C + L00I - L05V - L007
GSUR A00I + BLEND - L04W - L050 - L04X - C004
GSUR A00J + BLEND - L04V + L004 - L04Z + L050
GSUR A00K + BLEND + L04Z + L05C - L06J - L054
GSUR A00L + BLEND + L06J + L05D + L05F + L06I
GSUR A00N + BLEND + L04V + L052 + L06H + L05A
GSUR A00O + BLEND - L06H - L06F + L05I + L059
GSUR A00P + BLEND + L04W + L03R + L058 - L052
GSUR A00R + BLEND + L04X + L054 + L056 + L03S
GSUR A00S + BLEND - L058 + L05V + L05L + L06F
GSUR A00T + BLEND - L056 - L06I + L05S - L05Z
SETA wall l L05C
SETA wall l L059
SETA wall l L05A
SETA wall l L05D
SETA profil l L00C
SETA profil l L00E
SETA profil l L001
SETA profil l L00A
```

```
SETA profil 1 L009
SETA profil 1 L002
SETA profil 1 L00L
SETA in 1 L004
SETA out 1 L05F
SETA out 1 L05S
SETA out 1 L00G
SETA out 1 L00I
SETA out 1 L05I
SETA out 1 L05L
```

C.7 If and while demo

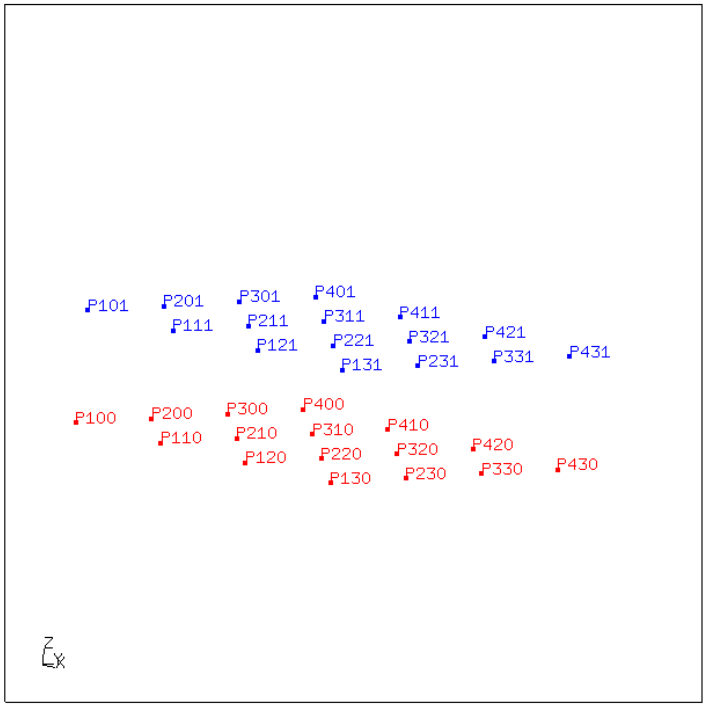


Figure 23: Result of If and while demo

The if and while commands can be nested. A demo which produces some points on the window follows:

```
text if&value&while demo
# def the leading letter of point names
```

```
valu vp P
# define the initial x value
valu vx 0.
# define parameters
valu v2 4
valu v3 1
# start loop:
while vx < v2
 valu vy 0.
 valu vz 0.
 seto S1
 while vy < v2
  # define the pnt coordinates
  valu vy + vy v3
  valu vy int vy
  valu vx int vx
  valu vz int vz
  # define the pnt name
  valu p1 & vp vy
  valu p1 & p1 vx
  valu p1 & p1 vz
  # generate the pnt
  pnt p1 vx vy vz
 endwhile
 setc
 valu vy 0.
 valu vz 1.
 seto S2
 while vy < v2
  valu vy + vy v3
  valu vy int vy
  valu vx int vx
  valu vz int vz
  valu p2 & vp vy
  valu p2 & p2 vx
  valu p2 & p2 vz
  pnt p2 vx vy vz
 endwhile
 setc
 valu vx + vx v3
 if vx == 1
   plot pa S1 r
 else
   plus pa S2 b
 endif
endwhile
```

```
send all fbd
# demo on how to write the content of variables to writedemo.txt
valu vx int vx
valu vy int vy
sys echo VX: vx VY: vx VZ: vz >| writedemo.txt
```

C.8 Data storage in a user dataset

```
# Example: Calculate normals of all free surfaces
# and write them to a new dataset
#
# switch on background mode (improves execution time)
asgn bg on
# get the number of surface nodes
seta n f all
comp n do
stack on
prnt se n
stack off
valu sum_nods pop
# calculate the normals
# and write all face-nodes to the stack (it writes them in inverse order)
stack on
norm n
stack off
#
# store the node numbers and values in array's (nod1 to nod<sum_nods>)
val n sum_nods
while n > 0
  valu cur_nod & nod n
  valu cur_val1 & val1_nod n
  valu cur_val2 & val2_nod n
  valu cur_val3 & val3_nod n
  val cur_nod pop
  val cur_val1 pop
  val cur_val2 pop
  val cur_val3 pop
  valu n - n 1
  valu n int n
endwhile
#
# create a new dataset
ds g NORMAL 3
#
# use the 'node' command to write data to the new dataset
# REMARK: 'n' has to be masked ('\') since is is already defined as a value
```

```
val \n sum_nods
while n > 0
  valu cur_nod & nod n
  valu cur_val1 & val1_nod n
  valu cur_val2 & val2_nod n
  valu cur_val3 & val3_nod n
  # code for cgx_2.16 and later:
  val \cur_nod cur_nod
  val \cur_val1 cur_val1
  val \cur_val2 cur_val2
  val \cur_val3 cur_val3
  # optional code for cgx_2.15:
  #val \cur_nod + cur_nod 0
  #val \cur_val1 + cur_val1 0
  #val \cur_val2 + cur_val2 0
  #val \cur_val3 + cur_val3 0
  valu cur_nod int cur_nod
  node cur_nod v cur_val1 cur_val2 cur_val3
  valu n - n 1
  valu n int n
endwhile
# set entity parameters
ds e nx 1 2 1
ds e ny 2 2 2
ds e nz 3 2 3
# finish
ds f
# switch off background mode
asgn bg off
# show it
ds 1 e 4
plot fv all
view vec
```

C.9 User File Parser

The following file will be parsed and two new datasets will be created. The nodes and elements must exist before execution!

```
# Modelname: oragl
# OrAg1 version: 19.7
## ONLINE OUTPUT ##
** CONTACT ELEMENT STATES @ amplitude=1.584893e-04
el.nr stick[%] slip[%] sep[%] FN mean[N] FN min[N] FN max[N] MU
208 0.0000 0.3594 0.6406 -1.7391E-02 -9.3037E-02 0.00005 6.00E-01
209 0.0703 0.2734 0.6562 -8.3892E-03 -4.6994E-02 0.00E00 6.00E-01
```

```
389 1.0000 0.0000 0.0000 -1.3890E+03 -1.3890E+03 -1.38E03 6.00E-01
390 1.0000 0.0000 0.0000 -6.9448E+02 -6.9450E+02 -6.94455 6.00E-01
391 1.0000 0.0000 0.0000 -1.3890E+03 -1.3890E+03 -1.38895 6.00E-01
392 1.0000 0.0000 0.0000 -6.9448E+02 -6.9452E+02 -6.94435 6.005-01

** CONTACT ELEMENT STATES @ amplitude=2.511886e-04
el.nr stick[%] s1ip[%] sep[%] FN mean[N] FN min[N] FN max[N] MU
208 0.0000 0.3594 0.6406 -2.7563E-02 -1.4745E-01 0.00005 6.00E-01
209 0.0703 0.2734 0.6562 -1.3296E-02 -7.4481E-02 0.00005 6.00E-01
389 1.0000 0.0000 0.0000 -1.3890E+03 -1.3890E+03 -1.38895 6.00E-01
390 1.0000 0.0000 0.0000 -6.9448E+02 -6.9451E+02 -6.94445 6.00E-01
391 1.0000 0.0000 0.0000 -1.3890E+03 -1.3890E+03 -1.38895 6.00E-01
392 1.0000 0.0000 0.0000 -6.9448E+02 -6.9454E+02 -6.94415 6.00E-01
```

The following code asks for the filename of the above listed data and stores the node related data in two new datasets each with seven entities.

```
valu string1 el.nr
valu string2 **
valu string3 CONTACT

valu file ? "provide oragl cstate filename:"

read file stack
stack on
prnt st si
stack off
valu sum_recs pop

val nn 0
while nn < sum_recs
  valu nn + nn 1
  valu nn int nn
  valu record & L nn
  val record pop
endwhile
stack free

valu nn 0
stack on
while nn < sum_recs
  valu nn + nn 1
  valu nn int nn
  valu record & L nn
  # REC record
  val record push
  valu arg1 pop
```

```
valu arg2 pop
if arg2 eq string3
  valu amplitude pop 4
  # AMP amplitude
  valu amplitude push =\
  valu amplitude pop 2
endif
if arg1 eq string1
  # found record arg1 string1
  # create a new dataset
  ds g CSTATE 7 amplitude
  valu cur_nod 0
  #
  while cur_nod ne string2
    # in while cur_nod ne string2
    valu nn + nn 1
    valu nn int nn
    if nn >= sum_recs
      # break nn sum_recs
      valu cur_nod string2
    else
      valu record & L nn
      val record push
      valu cur_nod pop
      valu arg1 pop
      valu arg2 pop
      valu arg3 pop
      valu arg4 pop
      valu arg5 pop
      valu arg6 pop
      valu arg7 pop
      node cur_nod v arg1 arg2 arg3 arg4 arg5 arg6 arg7
      seta CNODES \n cur_nod
      stack free
    endif
  endwhile
  #
  # set entity parameters
  ds e stick[%] 1
  ds e slip[%] 2
  ds e sep[%] 3
  ds e FNmean 4
  ds e FNmin 5
  ds e FNmax 6
  ds e MUE 7
  # finish
```

```
    ds f
    valu nn - nn 1
    valu nn int nn
  endif
endwhile
stack off
stack free
```

References

[1] OpenGL-Like Rendering Toolkit, from Brian Paul, http://www.mesa3d.org/

[2] OpenGL Utility Toolkit (GLUT), from Mark J. Kilgard

[3] CalculiX GraphiX (cgx), from Klaus Wittig, klaus.wittig@calculix.de

[4] NETGEN, unstructured mesher from Joachim Schoberl, https://sourceforge.net/projects/netgen-mesher/

[5] TETGEN, unstructured mesher from Hang Si, http://wias-berlin.de/software

[6] dolfyn, Open Source CFD code, http://www.dolfyn.net

[7] Duns, a two- and three dimensional cfd code, http://sourceforge.net/projects/duns/

[8] ISAAC, a two- and three dimensional cfd code, http://isaac-cfd.sourceforge.net

[9] OpenFOAM, a three dimensional cfd code, http://www.opencfd.co.uk

[10] Tochnog, a free fem-code, http://tochnog.sourceforge.net/

[11] Tutorial for CalculiX, from Dr. Guido Dhondt, http://www.dhondt.de/tutorial.html

[12] ImageMagick 5.1.0 00/01/01 Q:8 cristyg@mystic.es.dupont.com. Copyright: Copyright (C) 2000 ImageMagick Studio

[13] Mozilla Foundation, http://www.firefox.com

[14] S. A. Coons, 'Surfaces for computer-aided design of space forms'. Project MAC, MIT (1964). Revised to MAC-TR-41 (1967).

[15] mesh2d, unstructured 2D-mesher from B. Kaan Karamete, Ph.D, No URL Available

[16] Paul Dierckx, Curve and Surface Fitting with Splines, Oxford. University Press, 1993