NETGEN - 4.5

Joachim Schöberl and Robert Gaisbauer

January 11, 2010

Contents

Chapter 1

Getting Started

1.1 What is NETGEN

NETGEN is an automatic mesh generation tool for two and three dimensions. NETGEN is open source under the conditions of the LGPL. It comes as stand alone programme with graphical user interface, or as C++ library to be linked into an other application. NETGEN is available for Unix/Linux and Windows. NETGEN generates triangular or quadrilateral meshes in 2D, and tetrahedral or mixed tetrahedral, prismatic and pyramidal meshes in 3D. The input for 2D is described by spline curves, and the input for 3D problems is either defined by constructive solid geometries (CSG), see Chapter **??**, by the standard STL, IGES, STEP file formats or by the OpenCascade BREP file format. NETGEN contains modules for mesh optimization and hierarchical mesh refinement. Curved elements are supported of arbitrary order.

1.2 The history of NETGEN

The NETGEN project was started 1994 in the master's programme of Joachim Schöberl, under supervision of Prof. Ulrich Langer, at the Department of Computational Mathematics and Optimization, University Linz, Austria. Its further development was supported by the Austrian science Fund "Fonds zur Förderung der wissenschaftlichen Forschung" (http://www.fwf.ac.at) under projects P 10643-TEC and SFB 1306. The current home of NETGEN is the Start project "hp-FEM" (http://www.hpfem.jku.at) granted by the FWF. Special thanks go to

- Robert Gaisbauer: High order curved elements, OpenCascade interface
- Hannes Gerstmayr: Meshing of STL geometry

1.3 How to receive NETGEN

NETGEN is available from the WEB at

http://www.hpfem.jku.at/netgen

You find there source code releases for Linux/Unix/Windows, as well as compiled versions for Windows. You can use CVS access to receive the most up to date version.

1.4 Installing NETGEN

1.4.1 Installing NETGEN for Unix/Linux

To install NETGEN on Unix/Linux you will download the source code and compile it yourself. You need the following libraries:

- The 3D visualization library **OpenGL**. It comes with most systems with hardware graphics. The free software version mesagl is available from http://www.mesa3d.org.
- The graphical toolkit **TclTk** developed by John Ousterhout (available from http://www.scriptics.com/) and its extension **Tix** available from http://www.sourceforge.com) by Iam Lan. NETGEN has been tested with version TclTk 8.0 - TclTk 8.4 and Tix 4.6 - Tix 8.2

You can also download these packages from the NETGEN site.

To install NETGEN please move into the directory netgen. You set the Unix-variable MACHINE according to your machine/operating system, e.g.

```
setenv MACHINE LINUX
```

(in bash shell you type `export MACHINE=LINUX`). The Makefile includes the makefile-include

```
libsrc/makefile.mach.$(MACHINE)
```

Please create/modify the according file, (e.g. copy makefile.mach.LINUX to makefile.mach.SUN). Then you enter `make` to build the executable.

To make NETGEN globally available you just copy the binary "ng" to the global bin - directory. In difference to earlier versions, it needs no additional files.

1.4.2 Installing NETGEN for Windows

NETGEN is available now for Windows in binary form. Download the zip - archive and unpack it with winzip. You can start the executable "netgen.exe".

1.4.3 Adding IGES/STEP file support via OpenCascade

NETGEN is capable of importing IGES and STEP geometry files. If you want to use this functionality you have to add the OpenCascade library to your NETGEN distribution.

OpenCascade is an open source 3D solid modelling library by OpenCASCADE S.A. You can either obtain it from http://www.opencascade.org (Linux and Windows) or you can download a precompiled package containing all necessary files from our web page (Linux only).

To compile NETGEN with OpenCascade for Windows just choose the project settings "Release (OCC)" and adjust the proper search paths.

For Linux adjust the directory search paths OCC_DIR, OCCINC_DIR and OCCLIB_DIR in the Makefile and in libsrc/makefile.inc. Then add -DOCCGEOMETRY to the CPLUSPLUSFLAGS2 in libsrc/makefile.mach.$(MACHINE). If you use OpenCascade version 5.2 also add -DOCC52 and -DHAVE_IOSTREAM to the CPLUSPLUSFLAGS2.

1.4.4 Testing NETGEN

Please start NETGEN by entering "ng" or clicking the "netgen.exe" icon.

A white window with menu items should appear. Please load a geometry file by selecting "File -> Load Geometry", choose e.g. examples/cube.geo. Then press the button "Generate Mesh". By keeping pressed the left, middle or right button of your mouse you can rotate, move or zoom the object. With "File -> Export Mesh" you can save the mesh file.

Chapter 2

Constructive Solid Geometry (CSG)

The CSG input format is a useful geometry format for small and medium size geometries. One defines the geometry by writing an ASCII file in a text editor.

The geometry is defined by the Eulerian operations (union, intersection and complement) from primitives. A complete list of availabe primitives is given in Section **??**.

The following input describes a cube:

```
# A cube
algebraic3d
solid cube = orthobrick (0, 0, 0; 1, 1, 1);
tlo cube;
```

Lines starting with # are comment lines. Every CSG file must contain the keyword `algebraic3d` before any non-comment line. The keyword `solid` defines a named solid, here the solid *cube* is defined. A solid is defined by the Eulerian operations applied to primitives. Here, the solid is just the primitve defined by `orthobrick`. This is a brick parallel to the axis, specified by the minimal x, y, and z coordinates, and the maximal x, y, and z coordinates. The present definition gives the cube $[0,1]^3$. Finally, the definition `tlo cube` declares the solid *cube* as a top-level-object, what is necessary for meshing.

Please start netgen with the geometry file above by entering

ng cube.geo

Instead, you can also load the geometry from the file menu. You will see a blue cube, which you can rotate by keeping the left mouse button pressed. Pressing the big **generate mesh** button will result in a (very coarse) mesh of that cube.

Instead of using the primitive `orthobrick`, one can also specify a cube by intersecting six halfspaces (called planes). Each primitive `plane` is given by an

arbitrary point in the plane, and a outward vector, not necessarily a unit vector. The six halfspaces are intersected by the keyword `and`. The following input gives an equivalent result:

```
# A cube
algebraic3d
solid cube = plane (0, 0, 0; 0, 0, -1)
         and plane (0, 0, 0; 0, -1, 0)
         and plane (0, 0, 0; -1, 0, 0)
         and plane (1, 1, 1; 0, 0, 1)
         and plane (1, 1, 1; 0, 1, 0)
         and plane (1, 1, 1; 1, 0, 0);
tlo cube;
```

To drill a hole through the cube, we will intersect the cube and the complement of a cylinder. A cylinder is defined by two points on the central axis, and the radius. Please note, a cylinder is understood as an infinitely long cylinder (although the visualization may suggest a finite cylinder):

```
# cube with hole
algebraic3d
solid cubehole = orthobrick (0, 0, 0; 1, 1, 1)
         and not cylinder (0.5, 0.5, 0; 0.5, 0.5, 1; 0.1);
tlo cubehole;
```

Like `and` denotes the intersection, `or` denotes the union:

```
solid cubeball = orthobrick (0, 0, 0; 1, 1, 1)
        or sphere (0, 0, 0; 0.5) -maxh = 0.2;
```

The flag `-maxh=0.2` assignes the maximal mesh size of 0.2 to the solid. The current version, NG4.1, uses the mesh size assigned to the main solid of the top-level-object for the domain. Future version will contain more possibilities to define mesh-sizes for parts of a domain.

It is possible to define geometries with several sub-domains, simply by declaring several tlos:

```
algebraic3d
solid cube = orthobrick (0, 0, 0; 1, 1, 1);
solid cyl = cylinder (0.5, 0.5, 0; 0.5, 0.5, 1; 0.1);
solid dom1 = cube and not cyl;
solid dom2 = cube and cyl;

tlo dom1 -col=[0,0,1] -transparent;
tlo dom2 -col=[1,0,0];
```

This example show also solid trees involving previously defined named solids. Top-level-objects can be assigned a color specified by the amount of red, green and blue (RGB) values. The flag `-transparent` makes the solid appear transparent.

It is possible to specify bounday condition numbers for individual surfaces of a solid. The flag `-bc` assignes the bc to all surfaces of that solid-tree. If several flags are given the one closest to the leaves of the tree dominates. The following file defines a cube, with $bc = 1$ at the bottom, $bc = 2$ at the top, and $bc = 3$ for all other surfaces:

```
algebraic3d

solid bottom = plane (0, 0, 0; 0, 0, -1) -bc=1;
solid top    = plane (1, 1, 1; 0, 0,  1) -bc=2;
solid cube = bottm and top
         and plane (0, 0, 0; 0, -1, 0)
         and plane (0, 0, 0; -1, 0, 0)
         and plane (1, 1, 1; 0, 1, 0)
         and plane (1, 1, 1; 1, 0, 0) -bc=3;

tlo cube;
```

2.1 Available Primitives

NETGEN 4.1 supports the following primitives:

1. A halfspace, i.e., a plane and everything on one side of it, given by an arbitrary point $p = (p_x, p_y, p_z)$ in the plane and an outside normal vector $n = (n_x, n_y, n_z)$, not necessarily a unit vector:

   `plane (` p_x, p_y, p_z ; n_x, n_y, n_z `)`

2. A cylinder of infinite length, given by two points $a = (a_x, a_y, a_z)$ and $b = (b_x, b_y, b_z)$ on the central axis and the radius r:

   `cylinder (` a_x, a_y, a_z ; b_x, b_y, b_z ; r `)`

3. A sphere, given by the center $c = (c_x, c_y, c_z)$ and the radius r:

   `sphere (` c_x, c_y, c_z ; r `)`

4. An elliptic cylinder, given by the point $c = (c_x, c_y, c_z)$ on the main axis, and the vectors v and w of the long and short axis of the ellipse, respectively:

   `ellipticcylinder (`c_x, c_y, c_z ; v_x, v_y, v_z ; w_x, w_y, w_z`)`

5. An ellipsoid, given by the center $c = (c_x, c_y, c_z)$, and the vectors u, v and w of the main axis of the ellipsoid:

   `ellipsoid (`c_x`,` c_y`,` c_z `;` u_x`,` u_y`,` u_z`;` v_x`,` v_y`,` v_z `;` w_x`,` w_y`,` w_z`)`

6. A cone is given by two points on the central axis and the two corresponding radii. It is not possible to mesh the top of the cone yet, it must be cut off.

   `cone (` a_x`,` a_y`,` a_z `;` r_a`;` b_x`,` b_y`,` b_z `;` r_b `)`

7. A orthobrick is a brick parallel to the coordinate axis. It is specified by two opposite corner points $a = (a_x, a_y, a_z)$ and $b = (b_x, b_y, b_z)$:

   `orthobrick (` a_x`,` a_y`,` a_z `;` b_x`,` b_y`,` b_z `)`

8. A polyhedron is defined by a set of triangles forming a closed polyhedron. First, a set of points is defined, then the triangles are given by point indices. The triangles must be oriented counter-clockwise when looking onto the object. The following polyhedron describes a tetrahedron:

```
algebraic3d
solid poly = polyhedron (0,0,0; 1,0,0; 0,1,0; 0,0,1 ;;
                         1,3,2 ; 1,4,3; 1,2,4 ; 2,3,4 );
tlo poly;
```

2.2 Extended features

2.2.1 Singular points and edges

2.3 Known problems and work-arounds

2.3.1 Interfaces

A airdomain with two connected interior parts may be described by

```
algebraic3d

solid cube = orthobrick (0, 0, 0; 1, 1, 1);
solid part1 = orthobrick (0.2, 0.2, 0.2; 0.5, 0.8, 0.8);
solid part2 = orthobrick (0.5, 0.2, 0.2; 0.8, 0.8, 0.8);
solid air = cube and not part1 and not part2;

tlo air;
tlo part1;
tlo part2;
```

The problem is, that a domain is an open domain. Thus, the domain *air* is not only the outer part, but also the interface between *part1* and *part2*. The result is unspecified. To fix this problem, one can define the *air*-domain by cutting out one big brick:

```
solid air = cube and not othrobrick  (0.2, 0.2, 0.2; 0.8, 0.8, 0.8);
```

2.3.2 Degenerated edges

Degenerated edges are found sometimes, but some still cause troubles. A sphere on top of a cylinder my be described by:

```
solid cyl = cylinder (0, 0, 0; 1, 0, 0; 0.5)
        and plane (0, 0, 0; -1, 0, 0)
        and plane (1, 0, 0;  1, 0, 0);
solid main = cyl or sphere (1, 0, 0; 0.5);
tlo main;
```

The edge is a degenerated one. A work-around is to split the domain (artificially) into two non-degenerated parts:

```
solid cyl = cylinder (0, 0, 0; 1, 0, 0; 0.5)
        and plane (0, 0, 0; -1, 0, 0)
        and plane (1, 0, 0;  1, 0, 0);
solid hemisphere = sphere (1, 0, 0; 0.5) and not plane (1, 0, 0; -1, 0, 0);
tlo cyl;
tlo hemisphere;
```

Chapter 3

Other Geometry Formats

3.1 Using IGES/STEP Geometries

IGES and STEP are standard exchange formats for CAD files. Contrary to the STL format, IGES and STEP deliver an exact representation of the geometry and do not approximate it. In order to use IGES/STEP geometries you have to install NETGEN with the OpenCascade Geometry Kernel as described in **??**. Most solid modellers can export IGES or STEP files. However, often these files are not as exact as a mesher needs them to be. So is meshing fails, try repairing the model via **IGES/STEP Topology Explorer/Doctor** described in **??**.

3.2 Using STL Geometries

STL is a standardized file format to describe (approximate) geometies by triangulated surfaces. It is useful to describe complicated parts which are modeled with some CAD programmes. Also, some users have written their own (C) programmes to define STL geometries, where was not so easy to use the CSG format. The syntac of STL files is as follos

(not available yet. please figure out the syntax from the examples)

We found that many STL geometries have some difficulties. Some of them can be corrected (removed) by the **STL - Doctor**. Please see the corresponding manual pages (not available yet).

3.3 2D Spline Geometry

The extension for 2D spline geometry is ".in2d".

The boundary is given in terms of straight lines and of quadratic rational spline patches. A line is given by the two endpoints, a spline patch by 3 d'Boor

points. The patch is an elliptic arc from point 1 to point 3, such that the lines 1-2 and 2-3 are tangents.

It is possible to use different subdomains with this format.

This file format also supports a priori mesh grading. To the spline point i one adds a local refinement factor $\texttt{rp}_i$. Close to this point the mesh-size $h(x)$ is $\texttt{h}_{Glob}$ / $\texttt{rp}_i$. The global parameter `grading` describes how fast the mesh-size decreases. The gradient of the local mesh-size function $h(x)$ is bounded by $|\nabla_x h(x)| \leq \text{grading}^{-1}$ Also a refinement by a factor $\texttt{rs}_i$ ¿ 1 along the whole segment i is possible. The file looks like:

```
splinecurves2d
grading
np
x1 y1 rp1
...
xnp ynp rpnp
ns
dil1 dir1 [ 2 | 3 ] pi1,1 pi1,2 [ pi1,3 ] rs1
...
dilnl dirnl [ 2 | 3 ] pinl,1 pinl,2 [ pinl,3 ] rsnl
```

`np` is the number of points, `ns` the number of spline segments. Every segment starts with the domain numbers at the left and at the right sides of the segment. Domain number 0 is reserved for the exterior. Then the number of points and two or three point indices follow. Finally, the refinement factor along the line follows.

Chapter 4

Mesh, Solution and Demo Formats

You can export meshes to a couple of file formats. Some are self-defined, some other are standard formats. The self-defined are the followings:

4.1 Neutral Format

The neutral volume mesh format contains the following sections:

1. nodes
   After the number of nodes there follows a list of x, y, and z-coordinates of the mesh-nodes.

2. volume elements
   After the number of volume elements there follows the list of tetrahedra. Each element is specified by the sub-domain number, and 4 node indices. The node indices start with 1.

3. surface elements
   After the number of surface elements there follows the list of triangles. Each element is specified by the boundary condition number, and 3 node indices. The node indices start with 1.

4.2 Fepp Format 2D

The Fepp 2D format contains the following sections:

1. boundary segmetns
   After the number of boundary segments there follows a list of segments. Each segment is specified by the spline - patch number, and the two node indices. Counting starts with 1

2. domain elements
   After the number of domain elements there follows the list of elements. Each element is specified by the sub-domain number, the number of nodes (3 or 4) and the node indices. Counting starts with 1

3. nodes
   After the number of nodes there follows a list of x and y -coordinates of the mesh-nodes.

4. geometric information
   After the number of spline patches there follows a list of spline specifications. Each spline patch is given by the 6 coefficients of the descibing quadratic polynomial equation

$$c_1x^2 + c_2y^2 + c_3xy + c_4x + c_5y + c_6 = 0$$

4.3 Surface triangulaton file

One can export to and import from a surface mesh file. Its structure is as follows:

1. `surfacemesh`
   starts with that keyword

2. number of points
   point coordinates (x, y, z).

3. number of surface triangles,
   surface triangles oriented counter-clock wise when looking at the object, index starts from 1.

4.4 Solution File Format

The NETGEN software includes also a simple visualizer for finite element grid-functions. It supports scalar fields (e.g. temperature), and vector valued fields (flow velocities, mechanical deformations). The solution field is imported by the menu item File $->$ Import Solution. It is important to load the corresponding mesh in advance.

The format of the solution file is as follows. It consists of an arbitrary number of blocks of this structure:

1. `solution` *function-name* flags

   `solution` is the keyword, *function-name* is a string to refer to that functions. The supported flags are

(a) -size=s
number of entries (default: number of mesh-points)

(b) -components=c
number of components (default: 1). Mechanical deformations have 3 components.

(c) -type=[nodal,element,surfaceelement]
the grid-funciton has nodal values, or one value per volume element, or one value per surface element (default: nodal)

2. block of $size \times components$ values

Please try out to import the solution file 'tutorials/cube.sol' fitting to the mesh 'tutorials/cube.vol'.

4.5 Demo file format

Demo files (*.dem) are scripts that run a predefined animation of the displayed scene. One example demo script can be found in 'examples/test.dem'. Possible commands in demo scripts are:

- t = *time*;
  Sets the absolute time marker to *time* seconds.

- t += *seconds*;
  Increases the time marker *seconds* seconds.

- camerapos (...)
  Sets the 3D position of the virtual camera filming the display scene.

- camerapointto (...)
  Sets a 3D position where the camera is directed to.

- cameraup (...)
  Sets a 3D position that defines where the upside of the camera is directed to.

The last three commands all accept the same parameters, here only described fir 'camerapos'. You can choose between three different forms:

- camerapos (t_1: x_1,y_1,z_1);
  t_1 seconds after the last specified absolute time t, the camera position is moved to point (x_1,y_1,z_1) and stays there.

- camerapos (t_1: x_1,y_1,z_1; t_2: x_2,y_2,z_2);
  Beginning t_1 seconds after the last specified absolute time t, the camera position is moved linearly from point (x_1,y_1,z_1) to point (x_2,y_2,z_2), which is reached t_2 seconds after t.

- camerapos (t_1: x_1,y_1,z_1; t_2: x_2,y_2,z_2; t_3: x_3,y_3,z_3);
  Beginning t_1 seconds after the last specified absolute time t, the camera position is moved on an elliptic spline from point (x_1,y_1,z_1) to point (x_3,y_3,z_3), which is reached t_3 seconds after t. The lines 1-2 and 2-3 are tangents to the spline.

Here is an example script:

```
t = 0;
camerapos (0 :  -7,0,0; 10:  -5,0,0);
camerapos (10:  -5,0,0; 11:  -5,-5,0; 12:  0,-5,0);
camerapointto (11:  1,1,1);
```

Calling this script from the the 'Demo view' option in the 'File' menu does the following:

In the first 10 seconds, the camera moves linearly from point (-7,0,0) to point (-5,0,0), pointing to (0,0,0) (default value). In the next 2 seconds the camera moves on a spline to point (0,-5,0), while at time 11 the direction the camera points to jumps from (0,0,0) to (1,1,1).

Chapter 5

NETGEN operations

You can use netgen in interactive mode using its menus, or, you can run netgen in batch-mode using command line arguments.

5.1 Command line arguments

Command line arguments are specified as *-flag=value*.

- -help
  Prints the availabel command line arguments
- -geofile=filename
  Specifies geometry file. Is equivalent to *filename*, i.e., you can scip *-geofile=*.
- -meshfile=filename
  Mesh file will be stored in file *filename*.
- -batchmode
  Exit after mesh generation. Otherwise, the GUI will be started
- -V
  Verbose mode. Prints some additional information
- -verycoarse, -coarse, -moderate, -fine, -veryfine
  Mesh size control

Chapter 6

Using the Graphical User Interface

The NETGEN main window looks like:

It consists of the menuline and the button line at the top, the status line at the bottom, and the large drawing window. The menu items will be explained in **??**. The button line provides shot-cuts for common opteration:

- Quit
  Terminate NETGEN
- Generate mesh
  Performe mesh generation
- Stop Meshing
  Stop mesh generation
- Geometry/Edges/Mesh/Solution
  Switch between operation modes of visualization.
- Zoom all
  Zooms such that the whole visualization scene fits into the window.
- Center
  Center rotation and scaling at marked point, available only in mesh - visuailzation mode.

The status line shows information, namely

- Points
  Number of points in the mesh

- Elements
  Number of volume elements (3D) in the mesh
- Surf Elements
  Number of surface elements (3D) or inner elements (2d) in the mesh.
- Mem
  Used memory in the large memory arena
- Meshing Job, percentage Douriing mesh generation, the current job as well as the progress is displayed on the right side of the statu line.

The drawing window displays the geometry or the mesh. The view can be changed with the mouse:

- drag with left button pressed rotates the object,
- drag with middle button pressed moves the object,
- drag with right button pressed zooms the object.

The view can also be changed with the keyboard:

- cursor keys rotate the object
- shift + cursor keys move the object
- control + cursor keys zoom the object

When in Mesh - visualization scene, double clicking on triangles mark the surface. The point cursor is set.

When in Geometry - visualization scene with an IGES or STEP file loaded, double clicking on faces marks them and outputs their number. Useful for the *IGES/STEP Topology Explorer/Doctor*.

6.1 The NETGEN menu items

6.1.1 The menu item *File*

- Load Geometry
  Loads a geometry file into NETGEN. Supported file types are *.geo* (NETGEN's own 3D CSG geometry file format), *.in2d* (NETGEN's own 2D file format), *.stl*, *.bstl* (STL and binary STL file formats). If you have installed NETGEN together with the OpenCascade geometry kernel, there is additional support for *.iges* (IGES standard exchange format), *.step* (STEP standard exchange format) and *.brep* (OpenCascade's own file format).

- Save Geometry
  Save modified STL geometries as STL files and IGES or STEP geometries as IGES or STEP files. Modifications of geometries are possible through the STL doctor or the IGES/STEP Topology Explorer/Doctor.

- Recent Files
  Shortcuts for loading the last few geometry files.

- Load Mesh
  Loads a mesh from a *.vol* file.

- Save Mesh
  Save the mesh generated by NETGEN as *.vol* file.

- Merge Mesh
  Loads a mesh from a *.vol* file and merges it with the existing mesh.

- Import Mesh
  Load a mesh from *.mesh*, *.emt* (neutral file format), *.surf* (surface mesh format), *.unv* (universal format), *.emt* (olaf format) file.

- Export Mesh
  Saves the mesh in the file format chosen in *Export Filetype*.

- Export Filetype
  Choose the file format for exported meshes. Available file format are Neutral, Surface Mesh, DIFFPACK, TecPlot, Tochnog, Abaqus, Fluent, Permas, FEAP, Elmer, STL, VRML, and Gmsh.

- Save Solution
  Store the solution obtained by Finite Element Solver add-on NGSolve into *.sol* file.

- Import Solution
  Load a solution from a *.sol* file.

- Show Demo
  Loads and executes a demo script from a *.dem* file.

- Snapshot
  Perform a snapshot of the drawing window and save it to a *.jpg*, *.gif* or *.ppm* file.

- Save Options
  Stores the current meshing an visualization options into *ng.opt*. The next time NETGEN is started, these options are loaded.

- Quit
  Terminates NETGEN.

6.1.2 The menu item *Geometry*

- Scan CSG Geometry
  Generates surface triangulation for rendering.

- CSG Options

- CSG Properties
  Change the visualization parameters (color, visibility and transparency) for CSG top level objects.

- STL Doctor

- STL Info
  Prints some information on the the loaded STL geometry.

- IGES/STEP Topology Explorer/Doctor
  A complete tool for exploring and repairing the topological structure of IGES and STEP files as well as a repair tool for minor problems in the geometrical representation (shape healer). For further descriptions see **??**.

6.1.3 The menu item *Mesh*

- Generate Mesh
  Generates a mesh from the geometry. You can watch the mesh generation process if *parallel meshing thread* in the meshing options is checked. Otherwise NETGEN will not respond until the mesh is finished.

- Stop Meshing
  Terminates mesh generation.

- Meshing options
  Opens the meshing options dialog specified in section **??**.

6.1.4 The menu item *View*

- Zoom all
  Zooms such that the whole visualization scene fits into the window.

- Center
  Centers rotation and scaling at marked point, available only in mesh - visualization mode.

- x-y, y-x, x-z, z-x, y-z, z-y plane
  Sets the rotation such that you look onto the x-y, y-x, x-z, z-x, y-z, or z-y plane.

- Viewing Options
  Opens the viewing options dialog described in **??**.

- Clipping Plane
  Opens the clipping plane dialog described in **??**.

- Solution Data
  Opens the solution data visualization dialog described in **??**.

- Quality Plot
  Shows the element quality distribution histogram. Measure is volume scaled by edge length to the third. Optimal elements have quality 1.

- Help Line
  Displays a line at the bottom of the GUI showing interactive help for the pull down menues.

6.1.5 The menu item *Refinement*

- Refine uniform
  Refines the mesh by splitting all edges at the midpoint. ????

- Second Order
  ???????

- Validate Second Order

- High Order
  In you desire a curved mesh, this item performs the process of projecting the curved edges and faces onto the geometrical surface of the model. After mesh generation this procedure is done automatically. But if you change the element order for a finished mesh, you have to call this item afterwards. Remember, this item only make sense if you have set element order in the meshing dialog to a value greater than 1.

- Refinement Dialog
  Opens the refinement dialog described in **??**.

- Load Meshsize
  ??????????????

- MS from Surf Mesh
  ??????????????

- Singular point ms
  ??????????????

- Singular edge ms
  ??????????????

- Bisection
  ??????????????

- Z-Refinement
  ??????????????

- hp-Refinement
  ??????????????

- Split to Tets
  ??????????????

6.2 Meshing Options

6.2.1 The tab item *General*

Here you can set the most important parameters for getting a good result. In the **General** tab you can set the following parameters:

- Mesh granularity
  Choose between *very coarse*, *coarse*, *moderate*, *fine* and *very fine*, depending on how fine you want the final mesh to be. In fact this parameter is just a shortcut setting some of the parameters in the **Mesh Size** tab.

- First Step
  First step of the meshing process (in almost all cases set to *Analyze Geometry*)

- Last Step
  Choose between *Analyze Geometry*, *Mesh Edges*, *Mesh Surface*, *Optimize Surface*, *Mesh Volume*, *Optimize Volume*, depending on what type of mesh you want to generate (e.g. if you only need a mesh of the surface without generating volume elements, you best choose *Optimize Surface*).

- Print Messages
  Controls the amount of messages being output during mesh generation.

- Parallel meshing thread
  Active this checkbox if you want to view the growing mesh during the mesh generation process. If deactivated, you can not control NETGEN once the meshing process is started until it is finished.

- Second order elements
  ??????????????????????????????????

- Quad dominated
  NETGEN tries to mesh surfaces with quadritaterals. However, in general NETGEN will use triangles for about the last 5% to finish the surface meshing process.

- Invert volume elements
  ??????????????????????

- Invert surface elements
  ??????????????????????

- Element order
  The polynomial order of the edge, surface and volume elements created. Choose 1 for linear meshes and order greater that 1 for curved meshes. The meshing process starts always with a linear mesh and curves the edges and faces if required. If you change *Element order* for an already completed mesh, run *High order* from the *Refinement* menu afterwards to start the mesh curvature projection process.

- Large Memory
  Active NETGEN's own memory management in a large memory block.

6.2.2 The tab item *Mesh Size*

- max mesh-size
  Upper bound for the size of one element.

- mesh-size grading
  This parameter controls the transition in the mesh from finer to larger elements, i.e. from areas with a small local mesh size to areas with a large local mesh size. Enter a value between 0.1 and 1, where 0.1 means a smooth transition and 1 means an abrupt transition.

- mesh-size file
  ??????????????????????????

- Elements per curvature radius
  Controls the local mesh size against the local curvature of the geometry edges and surfaces. Slide between 0.2 and 5 elements per curvature radius. Larger values produce a finer mesh where the input geometry is curved stronger.

- Elements per edge
  Controls the local mesh size against the length of the edges of the input geometry. Slide between 0.2 and 5 elements per edge. Larger values produce a finer mesh where the input geometry has short edges.

- STL - chart distance
  ????????????????

- STL - line length
  ????????????????

- STL/IGES/STEP - close edges
  Controls the local mesh size against the distance of two edges of the input geometry. Slide between 0.2 and 8 elements per edge distance. Larger values produce a finer mesh along geometry edges which come close to each other.

- STL - surface curvature
  ????????????????

- STL - edge angle
  ????????????????

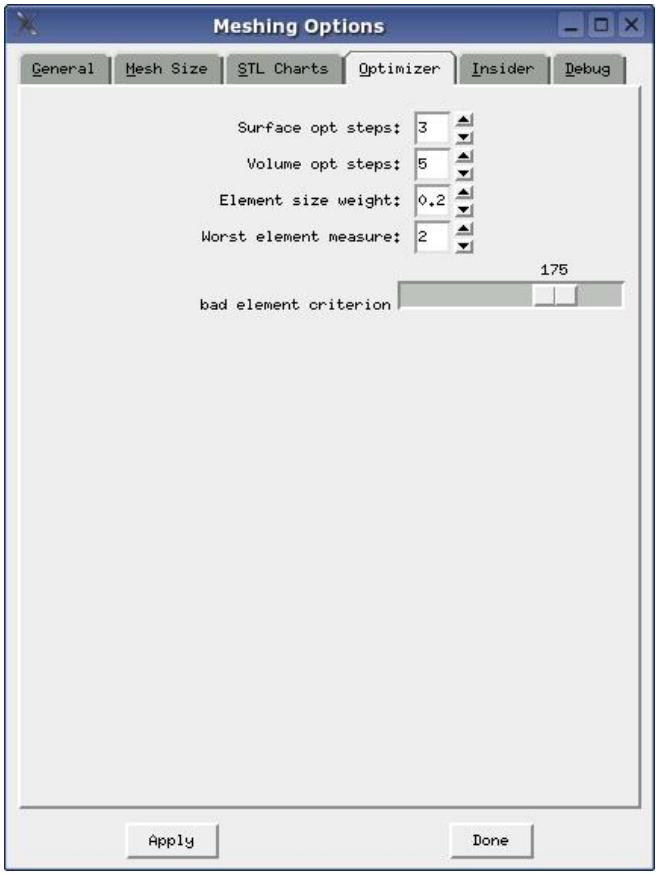
- STL - surface mesh curv
  ????????????????

- STL - recalc mesh size for surface optimization
  ??????????????
- Calc New H
  ??????????????

6.2.3 The tab item *STL Charts*

- Yellow Edges Angle
  ??????????????
- Edge Corner Angle
  ??????????????
- Chart Angle
  ??????????????
- Outer Chart Angle
  ??????????????

6.2.4 The tab item *Optimizer*



- Surface opt steps
  ??????????????
- Volume opt steps
  ??????????????
- Element size weight
  ??????????????
- Worst element measure
  ??????????????

6.2.5 The tab item *Insider*

- Use local mesh size
  ??????????????
- Use Delaunay
  ??????????????
- Check Overlapping
  ??????????????
- Do Blockfilling
  ??????????????
- Check Chart Boundary
  ??????????????

6.2.6 The tab item *Debug*

- Slow checks
  ??????????????
- Debugging output
  ??????????????
- Halt on exising line
  ??????????????
- Halt on overlap
  ??????????????
- Halt on success
  ??????????????

- Halt on no success
  ??????????????
- Halt on large quality class
  ??????????????
- Halt on segment
  ??????????????
- Halt on node
  ??????????????
- Halt on face
  ??????????????
- Show active meshing chart
  ??????????????
- P1, P2
  ??????????????
- Draw meshing, Meshing Testmode, Go on
  ??????????????

6.3 IGES/STEP Topology Explorer/Doctor

Whenever an IGES, STEP or BREP file is loaded, this dialog shows you the topological structure of the geometry. As we use the OpenCascade kernel for importing such files, the structure follows the rules and definitions of OpenCascade. There are the following entities in OpenCascade:

- Composite Solid
  Is a set of solids connected by their faces. It expands the notions of wire and shell to solids.
- Solid
  A part of space limited by shells. It is three dimensional.
- Shell
  A set of faces connected by their edges. A shell can be open or closed.
- Face
  Is part of a surface. Its geometry is constrained (trimmed) by contours. It is two dimensional.

- Wire
  A set of edges connected by their vertices. It can be an open or closed contour depending on whether the edges are linked or not.

- Edge
  A topological element corresponding to a restrained curve. An edge is generally limited by vertices. It has one dimension.

- Vertex
  A topological element corresponding to a point. It has zero dimension.

The tree in the Topology Explorer lets you see this structure, where the + and - signs means positive and negative orientation and the number next to it show of how many hierarchically lower elements an entity consists. Double clicking on the entities highlights them in the geometry visualization scene. if the checkbox *Zoom to highlighted entity* is marked, the visualization centers to the highlighted entity.

- Analyse Geometry
  checks for problematic entities. Very often, geometries contain very short edges that are not necessary for the geometrical description of the model. However, very short edges will result in t a (locally) very fine mesh, which most times is not necessary, too. *Analyse Geometry* lists the 20 shortest edges at the bottom of the tree, which gives you an idea of what edges can be omitted is the healing procedure described later.

- Healing tolerance
  Enter a tolerance value used for all the healing steps.

- Fix small edges
  Tries to replace all edges shorter than *healing tolerance* by prolongations of the neighboring edges. This is very useful if a very fine mesh is produced because of very short edges.

- Fix spot/strip faces
  Tries to replace very narrow or very small faces by expanding the neighboring faces.

- Sew faces
  Often, the topological structure of models ends at the face level. That means a model is only represented by its boundary faces, with no information on which faces are connected. Generating a surface mesh of such a geometry would produce indiviual meshes on the faces with no connection between them. The *sew faces* procedure sews faces with a distance closer than *healing tolerance* and connects them to shells.

- Make solids
  Remember that volume meshes can only be produced from solids. This procedure tries to make solids out of shells.

- Split partitions
  If solids are intersecting but this is not represented in the topology you can try to use this funtion. However, this is a very new algorithm which often does not work.

6.4 Viewing Options

6.4.1 The tab item *General*

- White background
  enabled: white background in visualization window, disabled: black background

- Draw coordinate cross
  if enabled, the x,y,z coordinate cross is visible in the lower left corner of the visualization window

- Draw color bar
  if enabled, a colored scale is visible at the top of the visualization window (whenever appropriate)

- Draw NETGEN logo
  if enabled, the text *Netgen 4.5* is visible in the lower right corner of the visualization window

6.4.2 The tab item *STL*

All these items are only working in the *Geometry* visualization scene, when an STL geometry file is loaded.

- Show STL-Triangles
  shows/hides the outlines of the triangles that describe the STL model

- Show Filled Triangles
  shows/hides filled triangles that describe the STL model

- Show Active Meshing-Chart

- Show Edges
  shows/hides geometrical edges found in the STL geometry. This feature works only after the *Analyze Geometry* step of the mesh generation process is run.

- Show Chart Triangles

- Show Faces

- Chart/Face number

- Chart/Face Offset

- Show Marked (Dirty) Triangles

- Show edge corner points

- Show touched triangle chart

- Draw meshed edges

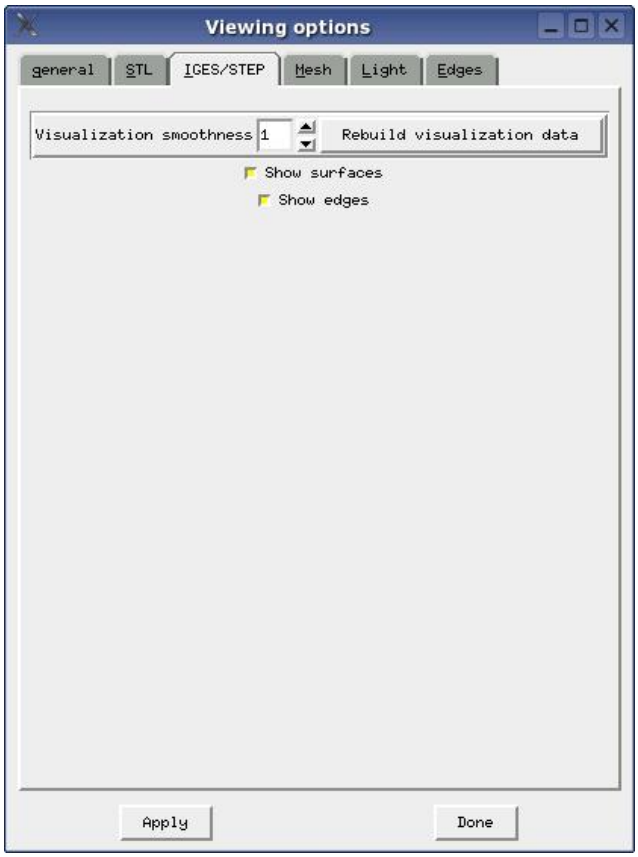
- Select with mouse

- Select triangle by number

- Show vicinity

- Vicinity size

6.4.3 The tab item *IGES/STEP*



All these items are only working in the *Geometry* visualization scene, when an IGES,STEP or BREP geometry file is loaded.

- Visualization smoothness
  Enter a value between 1 and 3 depending on how fine you want the visualization. Greater values mean finer visualization.

- Rebuild visualization data
  After changing the smoothness, press this button to rebuild the visualization mesh of the model.

- Show surfaces
  shows/hides the surfaces of the model

- Show edges
  shows/hides the edges of the model

6.4.4 The tab item *Mesh*

All these items are only working in the *Mesh* visualization scene.

- Set Center Point
  Centers the visualization scene around the specified mesh point
- Draw Element
- Show filled triangles
  Shows/hides surface mesh elements
- Show Triangle Outline
  Shows/hides outlines of the surface mesh elements
- Subdivision
  Curved elements are drawn by approximation them through linear elements. Subdivision 0 means drawing a curved triangle linearly, subdivision 1 means approximating the visualization by 4 linear triangles, subdivision 2 means approximating each of these 4 triangles by 4 triangles, i.e. approximate the curved element by 16 smaller, linear elements. Be aware that high numbers for subdivision mean finer, but slower visualization.
- Show edges
  Shows/hides the edge mesh
- Show bad elements
- Show surface of domain
- Show Point-numbers
  Shows/hides numbers of the mesh points
- Show Edge-numbers
  Shows/hides numbers of the mesh edges
- Show Face-numbers
  Shows/hides numbers of the mesh faces
- Show Element-numbers
  Shows/hides numbers of the mesh elements
- Show METIS Partition
  Shows the volume elements in different colors, corresponding to the partition computed by METIS.
- Show Tets in domain
  Shows/hides the tetrahedra in the specified domain (starting from 1). Domain 0 means all domains.

- Colored mesh size visualization
  Display a colored mesh. The color of an element correspond to the size of the element.
- Show pyramids
  Shows/hides pyramid volume elements.
- Show prisms
  Shows/hides prismatic volume elements.
- Show hexes
  Shows/hides hexahedral volume elements.
- Show identified points
- Shrink elements
  Displays the volume elements shrunk by a factor. 1 means no shrinkage.

6.4.5 The tab item *Light*

The items *Ambient Light, Diffuse Light, Specular Light, Material Shininess, Material Transparency* allow the adjustment of the style of the visualization.

6.4.6 The tab item *Edges*

All these items are only working in the *Edges* visualization scene.

- Show edges
  Shows/hides mesh edges
- Show points
  Shows/hides mesh points
- Show point nrs
  Shows/hides mesh point numbers
- Show CP Tangents
  Shows/hides tangents to the found cross points
- Show edge nrs
  Shows mesh edge numbers
- Set Center Point
  Centers the visualization scene around the specified mesh point
- SpecPoint Veclen
  Controls the length of the tangents

6.5 Clipping Plane

The clipping plane is a very useful tool for visualization. It is three dimensional plane dividing the visualization scene into two parts. One is visible and one is hidden.

- Normal x, y, z
  These sliders let you control the normal vector of the clipping plane.
- Distance
  Lets you control the distance of the clipping plane from the coordinate origin.
- Clip only domain nr
  ?????????????
- Do not clip domain nr
  ?????????????
- Enable clipping
  Enables/disables clipping

6.6 Solution Data Visualization

6.7 Refinement Dialog

Chapter 7

The Algorithms of NETGEN

NETGEN follows a top down strategy. It starts from computing the corner points (CSG only). Then, the edges are defined and meshed into segments (CSG and STL). Next, the faces are meshed by an advancing front surface mesh generator. After meshing, the faces meshes are optimized. Finally, the individual sub-domains are filled with tets. Therefore, a fast Delaunay algorithm generates most of the elements (about 98 percent). But often it fails for mesh the whole domain, then the slower back-tracking rule-base algorithm takes over. Finally, the volume is optimized by the usual node - movement, element swapping and splitting algorithms.

Chapter 8

Programming Interfaces

8.1 The nginterface

By means of the nginterface one's own simulation code can be included into the netgen environment. This is particular useful for FEM (FVM,BEM) code developers, since they may profit from the netgen preprocessing and postprocessing possibilities.

Please download the example NETGEN-add-on module *demoapp* and follow the instructions therein

8.2 The nglib

8.2.1 Introduction

The NETGEN mesh generation library *nglib* is available in C++ source code and can be compiled for Unix/Linux as well as Win95/98/NT and linked to one library file. The interface to the application programme is by the C language header file *nglib.h*.

The functionality of nglib is volume mesh generation by a domain given by a surface triangulation, and surface mesh generation from a domain described by an STL file (standard file format for geometries defined by triangle approximation). It can do mesh optimization as well as mesh refinement. It can generate 4 node tetrahedra and 10 node tetrahedrons (with quadratic shape functions). The local mesh size can be defined automatically by geometric features and/or by user specification.

8.2.2 The Header File

The interface file contains the following type definitions and function calls. All NETGEN types and functions start with `Ng`. Types and functions have capital

initial letters, constants are in capital letters.

8.2.3 Types and Constants

```
/// Data type for NETGEN mesh
typedef void * Ng_Mesh;

/// Data type for NETGEN STL geomty
typedef void * Ng_STL_Geometry;


// max number of nodes per element
#define NG_VOLUME_ELEMENT_MAXPOINTS 10

// implemented element types:
enum Ng_Volume_Element_Type { NG_TET = 1, NG_PYRAMID = 2, NG_PRISM = 3,
                              NG_TET10 = 4 };

// max number of nodes per surface element
#define NG_SURFACE_ELEMENT_MAXPOINTS 6

// implemented element types:
enum Ng_Surface_Element_Type { NG_TRIG = 1, NG_QUAD = 2,
                               NG_TRIG6 = 3 };


struct Ng_Meshing_Parameters
{
  double maxh;
  double fineness;   // 0 .. coarse, 1 .. fine
  int secondorder;
};

enum Ng_Result { NG_OK = 0,
                 NG_SURFACE_INPUT_ERROR = 1,
                 NG_VOLUME_FAILURE = 2,
                 NG_STL_INPUT_ERROR = 3,
                 NG_SURFACE_FAILURE = 4 };
```

`Ng_Mesh` is a data type representing a NETGEN mesh. `Ng_STL_Geometry` represents an STL geometry. One can operate on these data structures by the functions defined below. NETGEN can (now and/or in future) work with various

element types defined by generic constants. Several parameters can be specified in the `Ng_Meshing_Parameters` structure for volume and/or surface mesh generation. The result of NETGEN functions is of type `Ng_Result`.

8.2.4 Initialization

Please call these functions before using netgen functions and after using netgen functions, respectively:

```
// initialize, deconstruct NETGEN library:
void Ng_Init ();
void Ng_Exit ();
```

8.2.5 Mesh access

NETGEN meshes can be processed by the means of the following functions. A mesh contains nodes, surface elements and volume elements. Counting starts from 1.

```
// Generates new mesh structure
Ng_Mesh * Ng_NewMesh ();
void Ng_DeleteMesh (Ng_Mesh * mesh);

// feeds points, surface elements and volume elements to the mesh
void Ng_AddPoint (Ng_Mesh * mesh, double * x);
void Ng_AddSurfaceElement (Ng_Mesh * mesh, Ng_Surface_Element_Type et,
                           int * pi);
void Ng_AddVolumeElement (Ng_Mesh * mesh, Ng_Volume_Element_Type et,
                          int * pi);

// ask for number of points, surface and volume elements
int Ng_GetNP (Ng_Mesh * mesh);
int Ng_GetNSE (Ng_Mesh * mesh);
int Ng_GetNE (Ng_Mesh * mesh);


//  return point coordinates
void Ng_GetPoint (Ng_Mesh * mesh, int num, double * x);

// return surface and volume element in pi
Ng_Surface_Element_Type
Ng_GetSurfaceElement (Ng_Mesh * mesh, int num, int * pi);
```

```
Ng_Volume_Element_Type
Ng_GetVolumeElement (Ng_Mesh * mesh, int num, int * pi);
```

Mesh Generation

The user can specify the mesh size function by the global parameter maximal mesh size, and can additionally restrict the mesh size in points or cubes. The function `Ng_GenerateVolumeMesh` generates the volume mesh starting from the surface.

```
// Defines MeshSize Functions
void Ng_RestrictMeshSizeGlobal (Ng_Mesh * mesh, double h);
void Ng_RestrictMeshSizePoint (Ng_Mesh * mesh, double * p, double h);
void Ng_RestrictMeshSizeBox (Ng_Mesh * mesh, double * pmin, double * pmax, double

// generates volume mesh from surface mesh
Ng_Result Ng_GenerateVolumeMesh (Ng_Mesh * mesh, Ng_Meshing_Parameters * mp);
```

8.2.6 STL Geometry

A STL geometry can be read from a STL file (ASCII or binary), or can be assembled by providing triangle by triangle. Either, the user can specify the edges of the geometry, or netgen can define edges by `Ng_STL_MakeEdges` by using an angle criterium.

```
// loads geometry from STL file
Ng_STL_Geometry * Ng_STL_LoadGeometry (char * filename, int binary = 0);

// generate new STL Geometry
Ng_STL_Geometry * Ng_STL_NewGeometry ();

// fills STL Geometry
// positive orientation
// normal vector may be null-pointer
void Ng_STL_AddTriangle (Ng_STL_Geometry * geom,
                         double * p1, double * p2, double * p3, double * nv);

// add (optional) edges:
void Ng_STL_AddEdge (Ng_STL_Geometry * geom,
                     double * p1, double * p2);


// after adding triangles (and edges) initialize
```

```
Ng_Result Ng_STL_InitSTLGeometry (Ng_STL_Geometry * geom);

// automatically generates edges:
void Ng_STL_MakeEdges (Ng_STL_Geometry * geom);

  // generates mesh, empty mesh be already created.
Ng_Result Ng_STL_GenerateSurfaceMesh (Ng_STL_Geometry * geom,
                                      Ng_Mesh * mesh,
                                      Ng_Meshing_Parameters * mp);
```

8.2.7 Programming Example

The File *ngcore.cc*, see Appendix A, is a simple application using the netgen volume mesh generator. First, the surface mesh is read from a file containing point coordinates and surface triangles (see e.g. file *cube.surf*). The volume mesh generate is called, and the volume mesh is written to the standard output, see file *cube.vol*.