TOCHNOG User's manual - a free explicit/implicit FE program

Dennis Roddeman

February 11, 2001

Contents

1 Basic information

1.1 Introduction

TOCHNOG is a free finite element program.

Use the email address below for error reports, questions, wishes, etc. Please enclose a small input file if you report an error.

dennis.roddeman@feat.nl

1.2 Legal Issues

The GNU GENERAL PUBLIC LICENSE applies. It specifies COPYING conditions and a NO WARRANTY statement. It is included in the last chapter of this manual, and furthermore as a separate COPYING file in each distribution.

Realize that both my own written code and contributed code by others is owned (via copyright) by Dennis Roddeman. I do this with the explicit intention to make the code freely available under the Gnu Public License. It is setup this way because of logistics issues: legal decisions or modifications to the code would become completely unworkable if there were mixed copyrights. However, I want to credit and give thanks to those people that make the effort of creating code by maintaining a 'thanks to ...' section in the homepage. You can also add 'Programmed by ...' comments in the code that you contribute. If this is unacceptable for you, then don't contribute code.

1.3 Distribution files and installation

After installation of the source distribution file from the homepage you will find in the tochnog directory the following:

- src/tochnog.h
  Header file for C++ routines.
- src/makefile
  Compiles the program. Some precompiled executables are available at the download section of our home page. If you have another computer system then you need to compile yourself. Either use
  - make alpha (one-processor dec unix for alpha processors).
  - make alpha_parallel (multi-processor dec unix for alpha processors).
  - make hp (hp computers)
  - make linux (multi-processor linux version)
  - make linux_old (one-processor linux version)
  - make sgi (silicon graphics)
  - make sparc (sun solaris with sparc processors)
  - make sparc_parallel (for multi-processor sun solaris version)
  - make os2 (for the IBM OS2 platform)
  - nmake visual_cpp (for Microsoft windows; you need the visual c++ compiler)
  - nmake visual_cpp_parallel (for Microsoft windows; you need the visual c++ compiler)

- make borland_cpp (for Microsoft windows; you need the borland c++ compiler)

  where linux is for recent linux multi-processor platforms. The switch 'linux_old' is reported to compile correctly also on NeXT computers. If you don't have the commercial Borland or Visual compiler for Microsoft windows, you can try the free Cywin32 compiler (gcc compiler fow MS win32; see http://sourceware.cygnus.com) which is reported to compile correctly. The gcc compiler sometimes gives a wrong executable with the -O3 optimisation in the makefile (abvious, because tochnog will not run correctly at all); if so, change the optimisation back to -O.

- src/*.cc
  These are C++ routines.
- gid/*/*
  Files for the Tochnog graphical layer inside the GID pre- and postprocessor. See the separate Tochnog-GID manual on how to use these files.
- test/makefile
  Runs test input files on Unix. Use
  - make small
  - make large
  - make very_large
  - make all
- test/*.dat
  Data files.
- COPYING
  GNU GENERAL PUBLIC LICENSE
- tools/*.awk,cc
  These are simple translation programs between Tochnog and others (FE programs, pre-post)
- tools/tnwget.bat
  Script to load automatically tochnog distribution from internet. Needs the wget program.

After installation of the Latex distribution file from the homepage you will find in the tn directory the following:

- doc/tnu.tex
  User's documentation in Latex format.
- doc/tnp.tex
  Programmer's documentation in Latex format.
- doc/ps/*.ps
  Postscript figures in Latex documentation.

After installation of the html distribution file from the homepage you will find in the tn directory the following:

- doc/tnu/*
  User's documentation in html format. The entry file is tnu.html (or sometimes tnu.htm).
- doc/tnp/*
  Programmer's documentation in html format. The entry file is tnp.html (or sometimes tnp.htm).

- doc/tngid/*
  Tochnog-GID documentation in html format. The entry file is tngid.html (or sometimes tngid.htm).

To create a soft link on Unix to the Tochnog executable use:

- ln -s $HOME/tochnog/src/tochnog $HOME/bin/tochnog

If you put your bin directory in your PATH, you can use the Tochnog executable everywhere. As an alternative, you can move the executable to any directory which is part of your PATH.

1.4 How to perform a calculation and how to get started

Create an input file, e.g. **problem.dat**. The default input file is **tn.dat**, which will be used if no other input file is specified. Thus the command **tochog** or **tochnog tn.dat** yields output on the screen while **tochnog tn.dat > tn.out** redirects the output to a file. On a Unix system you can run the job in the background with **tochnog tn.dat > tn.out &**. On a Microsoft windows system you need to run from a Dos shell.

Use the **condif1.dat** test to get started.

- Copy **condif1.dat** to **tn.dat**.
- Use your favorite editor to open the file **tn.dat** and study it.
- Change **echo** to **-yes**.
- Remove the parentheses (...) surrounding the **control_print** statement and save the file.
- Run by typing **tochnog** or **tochnog tn** or **tochnog tn.dat**.
- Study the output on the screen.
- Study the **tn.log** file.
- Study the **tn.dbs** file. It contains the database after the calculation, and is an input file itself!

Afterwards do the same for the test **repeat1.dat**. It is simple, but shows some of the advanced features of TOCHNOG. Finally look at the test **repeat2.dat**. After that you understand much of the program.

Finally, read at least once the start of the data part section.

1.5 Plotting programs

You can use **GID** both for preprocessing (mesh generation) and postprocessing (plotting). An extension of GID is available by the name Tochnog-GID, see the homepage. The use of **GID** is recommended.

With **gnuplot** you can plot files resulting from **control_print_history** and **control_print_data_versus_data**. Also any other x-y plotting program can be used for such files.

Some other plotting programs can be also be used; see the **control_print_*** records.

See the links sections at our homepage where to get these programs.

1.6 Space discretization, time discretization

The computational domain is divided into finite elements. The elements connect at nodes. Either one-dimensional (1D), two-dimensional (2D), three-dimensional (3D) or axi-symmetrical (2D) domains can be used. All fields (velocities, stresses, etc.) are assumed to be continuous over the domain! This differs from common finite element programs where quantities like stresses, etc. can show jumps over element borders. This continuity gives a more diffuse but at the same time a more stable numerical scheme. Furthermore it has the advantage that all fields can be stored in the nodes (and not in integration points as in common finite element programs); this leads to a relatively easy implementation of remeshing and refining techniques.

Only first order in time equations are solved. Time derivatives are approximated with Euler backward time discretization.

1.7 Program capabilities

- Input

  Format free input. Words and no 'magic numbers' in rigidly defined columns are used.

  Boundary conditions can be imposed onto at geometrical entities, as well as onto elements and nodes.

- Output/plotting

  Output can be printed over user-specified geometrical objects (points, lines, quadrilaterals,...) as well as at nodes.

  The history of each variable, and for functions of variables, can be printed over user-specified geometrical objects as well as at nodes.

  Interface files can be created for both free and commercial post-processing programs.

- Finite elements

  1D, 2D and 3D. Tochnog mostly uses isoparametric elements. There are also springs, trusses and 2D beams however.

  Linear simplex elements (2 node bar, 3 node triangle, 4 node tetrahedron). A full family of first to fourth order bar, quadrilateral and brick elements.

- Mesh generation/refining/etc.

  Macro regions are automatically divided into finite elements.

  Local h-refinement (e.g. using residues in equations).

  Global h-refinement (more elements).

  Global p-refinement (polynomial refinement).

  Remeshing (using residues in equations or element shapes).

  Building a totally new mesh. Mapping from old state variables.

- Differential equations (materials)

  Convection-diffusion equation:

  - Temperature calculations.

  Fluids:

- Stokes and Navier-Stokes.

Solids:

- Elasticity (isotropy and transverse isotropy).

- Elasto-Plasticity (Von-Mises, Mohr-Coulomb, Gurson, etc.; plasticity surfaces can be arbitrarily combined).

- Hypo-Plasticity (Von-Wolfersdorf, cohesion, intergranular strains, pressure dependent initial void ratio).

- Damage.

- Thermal stresses.

- Hypoelasticity.

- Viscoelasticity.

- Viscoplasticity.

- Viscosity.

Ground water flow equation:

- Storage equation.

Wave equation.

- Accuracy information

  Residues in equations can be printed/plotted.

  Error estimates for all data (stresses, forces, temperatures, etc.)

- Interaction analysis

  Automatic fluid-solid interaction.

  Temperature effects on fluids, solids.

- Contact analysis

  Contact with and without friction.

  Frictional heat generation.

- Frames of description

  Lagrangian, Eulerian, and AEL (arbitrary Eulerian Lagrangian) calculations.

- Types of analysis

  Static, quasi-static and dynamic analysis.

- Parallelization

  The following functionality is parallized:

  - element nodal force calculation.

  - contact algorithm.

  - mapping of state variables when building a new mesh.

  - determination of boundary conditions.

  - iterative linear equations solver (diagonal preconditioned biconjugate gradient solver).

  - external PetSc linear equations solver (miscelaneous iterative solvers; MPI based parallelisation).

  - external SuperLU linear equations solver (direct LU solver; threads and MPI based parallelisation).

  - etc.

- Special features

  Automatic time-stepping (large steps for good iteration behavior, small steps for bad iteration behavior).

  Automatic distribution of tendons over linear finite elements (automatic embedment).

  Inverse modeling (estimation of model parameters).

  Restart possibility.

  Convection wiggle stabilization (both for low and high order elements).

- Costs

  Available for free. Including the source.

1.8 Files used by Tochnog

- Input file. By example **tn.dat**. The input file consists of an initialisation part (which unknowns should be solved, etc.) and a data part (elements, nodes, etc.).
- Runtime input file. By example **tn.run**. Use it to give Tochnog data records on the fly (while it is running).
- Log file **tn.log**. Contains log messages of calculations.
- Plot files. By example, **plotmtv1.dat**, **gmv001.inp**, ***.flavia.***, etc.
- Database file. By example, after the calculation with input file **condif1.dat** the database file **condif1.dbs** will be written. It contains everything (nodes, elements, solutions fields, etc.).
- Scratch files Don't use this name yourself. **tn.tmp**.

2 Equations

2.1 Convection and diffusion of heat

2.1.1 Convection-diffusion equation

$$\rho C(\dot{T} + \beta_i \frac{\partial T}{\partial x_i}) = k(\frac{\partial^2 T}{\partial {x_1}^2} + \frac{\partial^2 T}{\partial {x_2}^2} + \frac{\partial^2 T}{\partial {x_3}^2}) - aT + f$$

The primary unknown is the **condif_temperature** T. Further notation: ρ **group_condif_density**; C **group_condif_capacity**; x space coordinate; β_i **group_condif_flow** in i-direction; k **group_condif_conductivity**; a **group_condif_absorption**; f **force_element_volume**. Typical applications are heat conduction and heat conduction in a flow.

2.1.2 Convection to environment

$$q_c = \alpha_c(T - T_c)$$

Here q_c is the **condif_convection** heat flux, α_c is the convection coefficient and T_c is the environmental temperature for convection.

2.1.3 Radiation to environment

$$q_r = \alpha_r(T^4 - {T_r}^4)$$

Here q_r is the **condif_radiation** heat flux, α_r is the radiation coefficient and T_r is the environmental temperature for radiation.

2.2 Material deformation and flow

$$\rho \dot{v}_i = \frac{\partial \sigma_{ij}}{\partial x_j} + (1 - \beta T)\rho g_i - d \frac{\partial v_i}{\partial x_i}_{\text{no sum on i}} + f_i$$

Notations: ρ **materi_density**; v_i **materi_velocity** in i-direction; σ_{ij} **materi_stress** matrix; x space coordinate; β **group_materi_expansion_volume**; T (optional) **condif_temperature**; g_i **force_gravity**; d **group_materi_damping** coefficient; f_i **force_element_volume**. The equation is given for space coordinates following the material velocities v_i.

TOCHNOG allows you to build your favorite material, by adding separate contributions to the stresses σ_{ij}. In this way you can build solids or fluids or things in between. The separate contributions will be listed below. First two typical examples are given.

Nearly incompressible Navier Stokes:

...
materi_velocity
materi_stress
end_initia
...
group_type 0 -materi
group_materi_elasti_compressibility 0 1.0
group_materi_viscosity 0 1.2

Linear solid:

...
materi_velocity
materi_strain_total
materi_stress
end_initia
...
group_type 0 -materi
group_materi_elasti_young 0 1.e10
group_materi_elasti_poisson 0 0.2
group_materi_memory 0 -updated_without_rotation

2.2.1 Memory

The **-updated** Lagrange formulation

Deformations (i.e. the incremental deformation matrix F) refers to the previous time point. TOCHNOG decomposes the incremental deformation tensor with a polar decomposition into $F = RU$ with F the incremental deformation matrix, R the incremental rotation matrix and U the incremental stretch matrix. The incremental stretch matrix U is used to determine the incremental strain matrix $0.5(U + U^T) - I$ with I the identity tensor. The velocities between the two time points are the unknowns to be solved. Stresses are calculated from adding incremental stresses to the old stresses. Incremental stresses are caused by the incremental strain matrix and a rigid body rotation by the incremental rotation matrix of the old stresses.

The **-updated_without_rotation** Lagrange formulation

Deformations (i.e. the incremental deformation matrix F) refers to the previous time point. Any rigid body rotation between the two time points are neglected, so TOCHNOG decomposes the incremental deformation tensor with a polar decomposition into $F = U$ with F the incremental deformation matrix, and U the incremental stretch matrix. The velocities between the two time points are the unknowns to be solved. Stresses are calculated from adding incremental stresses to the old stresses. Incremental stresses are caused by the incremental strain matrix.

The **-total** Lagrange formulation

Deformations (i.e. the total deformation matrix F) refers to the time 0. TOCHNOG decomposes the total deformation tensor with a polar decomposition into $F = RU$ with F the total deformation matrix, R the total rotation matrix and U the total stretch matrix. The total stretch matrix U is used to determine the total strain matrix $0.5(U + U^T) - I$ with I the identity tensor. The displacements between the current time and time 0 are the unknowns to be solved. Stresses are calculated from the total displacements.

The **-total_piola** Lagrange formulation

Deformations (i.e. the total deformation matrix F) refers to the time 0. TOCHNOG decomposes the total deformation tensor with a polar decomposition into $F = RU$ with F the total deformation matrix, R the total rotation matrix and U the total stretch matrix. The total stretch matrix U is used to determine the total Green-Lagrange strain matrix $0.5(F^T F - I) = 0.5(U^T U - I)$ Stresses are calculated from the total displacements.

The **-total_linear** Lagrange formulation

Deformations (i.e. the total deformation matrix F) refers to the time 0. TOCHNOG neglects any rigid body rotations and uses linear engineering strains $0.5(F + F^T) - I$. The displacements between the time current time and time 0 are the unknowns to be solved. Stresses are calculated from the total displacements.

A remark on the total Lagrange models. Normally stresses are calculated from the total displacements (and thus the total strains). The old stresses are not used. This means that any initial stresses are neglected. This type of stress calculation for the total Lagrange models is used whenever **materi_strain_total** is not initialised. However, in case **materi_strain_total** is initialised, the difference between the total displacements (and thus strains) between two time points is used to determine incremental stresses, which are added to the stresses at the previous time point. And thus, in case **materi_strain_total** is initialised, the old stresses are used and any initial stresses are not neglected.

See also **group_materi_memory**.

2.2.2 Elasticity

The elastic stress rate is

$$C_{ijkl} \dot{\epsilon_{kl}}^{\mathrm{elas}}$$

where C_{ijkl} is the elastic modulus tensor (which is a doubly symmetric tensor: $C_{ijkl} = C_{jikl}$, $C_{ijkl} = C_{ijlk}$ and $C_{ijkl} = C_{jilk}$), and $\dot{\epsilon_{kl}}^{\mathrm{elas}}$ is the elastic strain rate. See the plasticity section for a definition of the elastic strain rate.

For an isotropic material

$$C_{0000} = C_{1111} = C_{2222} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}$$

$$C_{0011} = C_{0022} = C_{1122} = \frac{E\nu}{(1+\nu)(1-2\nu}$$

$$C_{0101} = C_{0202} = C_{1212} = \frac{E}{1+\nu}$$

with E **group_materi_elasti_young** modulus and ν **group_materi_elasti_poisson** ratio (the remaining non-zero moduli follow from the double symmetry conditions).

For a transverse isotropic material the material has one unique direction (think of an material with fibers in one direction). Here we take 'a' as the unique direction; 'b' and 'c' are the transverse directions. The material is fully defined by C_{aaaa}, C_{bbbb}, C_{aabb}, C_{abab} and C_{bcbc} and the unique direction in space (see **group_materi_elasti_transverse_isotropy**). The other non-zero moduli follow from $C_{cccc} = C_{bbbb}$, $C_{acac} = C_{abab}$, $C_{bbcc} = C_{bbbb} - 2C_{bcbc}$ and from the double symmetry conditions.

The nonlinear elasticity polynomials is a strain dependent model. In this model, the 'young's stiffness' modulus is made dependend of the size of the strains via a series of polynomials

$$E = E_0 + E_1\epsilon^1 + E_2\epsilon^2 + \ldots \tag{1}$$

where

$$\epsilon = \sqrt(\epsilon_{ij}\epsilon_{ij}) \tag{2}$$

with ϵ_{ij} the components of the strain matrix. The parameters E_0 etc. need to be specified in the **group_materi_elasti_young_polynomial** record.

The power law nonlinear elasticity is a stress dependent model which typically is used to model the elastic behavior of granular materials. It can be combined with plastic models, by example with the di Prisco plasticity model for soils, and with a poisson ratio.

In this model, the 'young's stiffness' modulus is made a function of the average stress state:

$$E = E_0(p/p_0)^\alpha \tag{3}$$

where p is the pressure. Furthermore, E_0 is the reference stiffness at reference pressure p_0, and α is a soil dependent power coefficient. The parameters E_0, p_0, and α need to be specified in the **group_materi_elasti_young_power** record.

The Lade nonlinear elasticity is a stress dependent model which typically is used to model the elastic behavior of granular materials. It can be combined with plastic models, by example with the di Prisco plasticity model for soils.

The stress rates are linked to the strain rates by the equation:

$$\dot{\epsilon_{ij}} = \frac{\partial W^2}{\partial\sigma_{ij}\partial\sigma_{hk}}\ \dot{\sigma_{hk}} \tag{4}$$

where the function W is

$$W = \frac{X^{1-\lambda}}{2B(1-\lambda)}$$

where

$$X = p^2 + R^* \mathrm{abs(s_{ij}s_{ij})}$$

with pressure $p = (\sigma_{11} + \sigma_{22} + \sigma_{33})/3$ and deviatoric stresses $s_{ij} = \sigma_{ij} - p\delta_{ij}$.

The model contains three user specified constants B, R, λ which need to be specified in the **group_materi_elasti_lade** record. B and λ are defined by means of an isotropic unloading test, and R by means of an unloading-standard-triaxial-compression test. For example for a loose sand $B = 1028$, $R = 0.25$, $\lambda = 0.28$. See [7] for the details.

The model cannot be used in combination with a poisson ratio.

2.2.3 Elasto-Plasticity

<u>Plastic strain</u>

In plastic analysis, the **materi_strain_elasti** rate follows by subtracting from the **materi_strain_total** rate the **materi_strain_plasti** rate

$$\dot{\epsilon_{ij}}^{\mathrm{elas}} = \dot{\epsilon_{ij}} - \dot{\epsilon_{ij}}^{\mathrm{plas}}$$

where the **materi_strain_total** rate is

$$\dot{\epsilon_{ij}} = 0.5(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i})$$

The **materi_strain_plasti** rate follows from the condition that the stress cannot exceed the yield surface. This condition is specified by a yield function $f^{\mathrm{yield}}(\sigma_{ij}) = 0$. The direction of the plastic strain rate is specified by the stress gradient of a flow function $\frac{\partial f^{\mathrm{flow}}}{\partial \sigma_{ij}}$. If the yield function and flow function are chosen to be the same, the plasticity is called associative, else it is non-associative.

Von-Mises is typically used for metal plasticity. Mohr-Coulomb and Drucker-Prager are typically used for soils and other frictional materials. *The plasticity models can freely be combined; the combination of the plasticity surfaces defines the total plasticity surface.*

First some stress quantities which are used in most of the plasticity models are listed.

Equivalent Von-Mises stress:

$$\bar{\sigma} = \sqrt{\frac{s_{ij}s_{ij}}{2}}$$

Mean stress:

$$\sigma_m = \frac{\sigma_{11} + \sigma_{22} + \sigma_{33}}{3}$$

Deviatoric stress:

$$s_{ij} = \sigma_{ij} - \sigma_m \delta_{ij}$$

Modified CamClay plasticity model

Here we give the equations of the Modified Cam Clay model (Roscoe and Burland, 1968, summarized e.g. by Wood, 1990, see [12]). All stresses are effective (geotechnical) stresses, i.e.compression is positive! Definitions of variables:

$$p = (\sigma_{11} + \sigma_{22} + \sigma_{33})/3$$

$$q = \{\frac{1}{2}[(\sigma_{11} - \sigma_{22})^2 + (\sigma_{22} - \sigma_{33})^2 + (\sigma_{33} - \sigma_{11})^2] + 3(\sigma_{12}^2 + \sigma_{23}^2 + \sigma_{31}^2)\}^{1/2}$$

The CamClay yield rule, which is also the flow rule, reads:

$$f = g = q^2 - M^2[p(p_0 - p)] = 0$$

M is a soil constant and p_0 is a history (hidden) variable which corresponds to the preconsolidation mean pressure. The hardening function, evolution, of p_0 reads:

$$dp_0 = \frac{p_0(1+e)d\varepsilon_v^p}{\lambda - \kappa}$$

in which

$$d\varepsilon_v^p = d\varepsilon_{11}^p + d\varepsilon_{22}^p + d\varepsilon_{33}^p$$

and λ and κ are user specified soil constants. Further e is the void ratio with the evolution equation:

$$de = -d\varepsilon_v(1 + e)$$

in which

$$d\varepsilon_v = d\varepsilon_{11} + d\varepsilon_{22} + d\varepsilon_{33}$$

The poisson ratio ν reads:

$$\nu = \frac{3K - 2G}{2G + 6K}$$

in which the elastic bulk modulus K is given by:

$$K = (1 + e)p/\kappa$$

and the Young's modulus E:

$$E = 2. * G * (1 + \nu)$$

in which G is a user specified soil constant, By using this ν and E the classical isotropic stress-strain law is used to calculate the stresses.

The soil constants M, κ, λ need to be specified in **group_materi_plasti_camclay**. The soil constant G, need to be specified in **group_materi_elasti_camclay_g**. For an alternative see **group_materi_elasti_camclay_poisson**. The history variables e, p_0 need to be initialized by **materi_history_variables 2** record (and given initial values in **node_dof** records).

Remark 1: An additional parameter N can be often found in textbooks on the Cam Clay model. We don't include it since it is linked to other model parameters via:

$$1 + e = N - \lambda \ln p_0 + \kappa \ln(p_0/p)$$

Remark 2: If you apply a geometrical linear analysis, see section 8.4, then also the calculation of de void ratio development is linearized, and so will contain some error as compared to the exact void ratio change. Hence for very large deformations, say above 10 percent or so, don't use such geometrical linear analysis.

Cap plasticity model

This model accounts for permanent plastic deformations under high pressures for granular materials. It is intended to be used in combination with shear plasticity models like Drucker-Prager, Mohr-Coulomb, etc.

First a deviatoric stress measure t and hydrostatic stress measure p are defined

$$t = \sqrt{3}\bar{\sigma}$$

$$p = -\sigma_m$$

See above for $\bar{\sigma}$ and σ_m. The yield rule for the **group_materi_plasti_cap** model reads:

$$f = \sqrt{(p - p_a)^2 + \left[\frac{Rt}{(1 + \alpha - \frac{\alpha}{cos\phi}} \right]^2} - R(c + p_a tan\phi)$$

Here c is the cohesion and ϕ is the friction angle which should be taken equal to the values in the shear flow rule which you use. The parameter p_a follows from

$$p_a = \frac{p_b - Rc}{1 + R\ tan\phi}$$

where the hydrostatic compression yield stress p_b is to be defined with an table of volumetric plastic strains ϵ_v^p versus p_b with $\epsilon_v^p = \epsilon_{11}^p + \epsilon_{22}^p + \epsilon_{33}^p$. As always, positive strain denote extension whereas negative strains denote compression.

Associative flow is used, so the flow rule is taken equal to the yield rule.

Summarizing the **group_materi_plasti_cap** model needs as input the cohesion c, the friction angle ϕ, the parameter α (typically 1. 10^{-2} up to 5. 10^{-2}), and a table ϵ_v^p versus p_b.

Compression limiting plasticity model

This **group_materi_plasti_compression** model uses a special definition for the equivalent stress

$$\bar{\sigma} = \sqrt{{\sigma_1}^2 + {\sigma_2}^2 + {\sigma_3}^2}$$

where σ_1, σ_2 and σ_3 are the first, second and third principal stress respectively. Each of these is only incorporated if it is a compression stress. The model now reads

$$\bar{\sigma} - \sigma_y = 0$$

This plasticity surface limits the allowed compression stresses.

Tension limiting plasticity model

This **group_materi_plasti_tension** model uses a special definition for the equivalent stress

$$\bar{\sigma} = \sqrt{{\sigma_1}^2 + {\sigma_2}^2 + {\sigma_3}^2}$$

where σ_1, σ_2 and σ_3 are the first, second and third principal stress respectively. Each of these is only incorporated if it is a tension stress. The model now reads

$$\bar{\sigma} - \sigma_y = 0$$

This plasticity surface limits the allowed tension stresses.

A simple model for concrete can be obtained as follows. Use **group_materi_plasti_tension** to limit the tension strength *ft*. Use **group_materi_plasti_compression** to limit the compressive strength *fc*. The tension strength could be softened to zero over an effective plastic strain κ of, say, 1 percent. The compressive strength could be softened to zero over an effective plastic strain κ of, say, 10 percent.

Another possibility for concrete is to combine **group_materi_plasti_tension** to limit the tension strength *ft*, and use the **group_materi_plasti_vonmis** model to limit the compressive strength *fc*.

di Prisco plasticity model

The di Prisco model is an non-associative plastic model for soils, which can be typically combined with the 'Lade elastic model'. This di Prisco model is a rather advanced soil model, which is explained in more detail in [4] and [5]. The yield rule reads:

$$f = 3\beta_f(\gamma - 3)\ln\left(\frac{r}{r_c}\right) - \gamma J_{3\eta^*} + \frac{9}{4}(\gamma - 1)J_{2\eta^*}$$

and the flow rule yields:

$$g = 9(\gamma - 3)\ln\left(\frac{r}{r_g}\right) - \gamma J_{3\eta^*} + \frac{9}{4}(\gamma - 1)J_{2\eta^*}$$

This is an anisotropic model in which the first and second invariant of the stress rate η^* are defined relative to the rotation axes χ.

$$r = \sigma_{ij}\chi_{ij}$$

$$J_{3\eta^*} = \eta^*_{ij}\eta^*_{jk}\eta^*_{ki}$$
$$J_{2\eta^*} = \eta^*_{ij}\eta^*_{ij}$$
$$\eta^*_{hk} = \sqrt{3}\frac{s^*_{hk}}{r}$$

where s^* follows from

$$s^*_{hk} = \sigma^*_{hk} - r\chi_{hk}$$

Further $r_g = 1$.

The history variables are χ_{ij} (rotation axes, 9 values), β (yield surface form factor), and r_c (preconsolidation mean pressure). The evolution laws for these history variables can be found in the papers listed above. The history variables χ_{ij} (9 values), β, r_c need to be initialized by the **materi_history_variables 11** record (and should be given initial values in **node_dof** records). In a normally consolidated sand with isotropic initial conditions $\chi_{ij} = \frac{\delta_{ij}}{\sqrt{3}}$, $\beta = 0.0001$ and r_c equals $\sqrt{3}$ times the means pressure.

The total model, yield rule and flow rule and evolution laws for history variables, contains a set of soil specific constants. In **group_materi_plasti_diprisco** you need to specify these constants. These constants are explained in more detail in the papers mentioned above, but here we give a short explanation. The constants $\hat{\theta}_c$, $\hat{\theta}_e$, ξ_c and ξ_e are linked to the dilitancy and the stress state during failure (standard triaxial compression and extension test in drained conditions). The constants γ, c_p, β_f and β^0_f are defined by means of the experimental curves (q-ϵ_{axial}, ϵ_{vol}-ϵ_{axial}) obtained by performing a standard compression test in drained conditions. Moreover, β_f, β^0_f and t_p can also be determined by means of the effective-stress path obtained by performing a standard triaxial compression test in undrained conditions.

A cohesion C can be also be introduced if required.

Finally b_p can determined from an isotropic compression test. For a loose sand $\hat{\theta}_c = 0.253$, $\hat{\theta}_e = 0.0398$, $\xi_c = -0.2585$, $\xi_e = -0.0394$, $\gamma = 3.7$, $c_p = 18.$, $\beta_f = 0.5$, $\beta^0_f = 1.1$, $t_p = 10.$, and $b_p = 0.0049$.

Drucker-Prager plasticity model

The **group_materi_plasti_druckprag** model reads

$$3\alpha\sigma_m + \bar{\sigma} - K = 0$$

$$\alpha = \frac{2\sin(\phi)}{\sqrt{3}(3-\sin(\phi))}$$

$$K = \frac{6c\cos(\phi)}{\sqrt{3}(3-\sin(\phi))}$$

Here c is the cohesion, which needs to be specified both for the yield function and the flow rule; by choosing different values non-associative plasticity is obtained.

Gurson plasticity model

The **group_materi_plasti_gurson** model reads

$$\frac{3\bar{\sigma}^2}{\sigma_y^2} + 2q_1 f^* \cosh(q_2 \frac{3\sigma_m}{2\sigma_y}) - (1 + (q_3 f^*)^2) = 0$$

Here f^* is the volume fraction of voids. The rate equation

$$\dot{f^*} = (1 - f^*) f^* \epsilon_{kk}^{\text{plas}}$$

defines the evolution of f^* if the start value for f^* is specified. Furthermore, q_1, q_2 and q_3 are model parameters.

HLC plasticity model

The **group_materi_plasti_hlc** Hau-Liu-Chang model is nearly similar to the Gurson's equation, and it is reads

$$\frac{3\bar{\sigma}^2}{\sigma_y^2} + f^*(+\frac{1}{m_{20}})m_1 \exp(\frac{3\sigma_m}{2\sigma_y}) - 1 = 0$$

Here f^* is the volume fraction of voids.

The rate equation

$$\dot{f^*} = (1 - f^*) f^* \epsilon_{kk}^{\text{plas}}$$

defines the evolution of f^* if the start value for f^* is specified. The variable m_{20} is a function of the porosity:

$$m_{20} = (m_{21} - m_{22} f^*) \exp(m_{21} \frac{\sigma_m}{\bar{\sigma}})$$

Furthermore, σ_y, m_{21}, m_{22}, m_{23} and m_1 are model parameters. The parameters can be calculated numerically by simple numerical simulations.

See http://www.tam.nwu.edu/wkl/paper/suhao-paper2.html for further info about the model.

Von-Mises plasticity model

The **group_materi_plasti_vonmis** model reads

$$\sqrt{3}\ \bar{\sigma} - \sigma_y = 0$$

where without hardening the yield value is fixed $\sigma_y = \sigma_{y0}$.

If however the **group_materi_plasti_vonmis_nadai** hardening law for Von-Mises plasticity is specified then

$$\sigma_y = \sigma_{y0} + C(\kappa\_0 + \kappa)^n$$

where C, κ_0 and n are parameters for the hardening law, and κ is the isotropic hardening parameter (see later). The parameter σ_{y0} is specified by **group_materi_plasti_vonmis**.

Modified Matsuoka-Nakai model plasticity model

The Matsuoka-Nakai model [8] is a perfectly plastic model thus the fixed yield surface represents the failure surface as well. The model is based on experimental results with soils and can be formulated in terms of three stress invariants

$$f = I_3 + \frac{\cos^2\phi}{9 - \sin^2\phi}\, I_1\, I_2 = 0$$

where

$$\begin{aligned} I_1 &= \mathrm{tr}(\sigma_{ij}) = \sigma_{11} + \sigma_{22} + \sigma_{33} = \sigma_1 + \sigma_2 + \sigma_3 = 3\sigma_m \\ I_2 &= \frac{1}{2}\left(\mathrm{tr}(\sigma_{ik}\sigma_{kj}) - I_1^2\right) = -\sigma_1\sigma_2 - \sigma_2\sigma_3 - \sigma_3\sigma_1 \\ I_3 &= \det(\sigma_{ij}) = \sigma_1\sigma_2\sigma_3 \end{aligned}$$

σ_1, σ_2 and σ_3 are the principal stresses (all stresses are effective; compressive stresses are negative). The parameter ϕ is equal to the angle of internal friction in axisymmetric (triaxial) compression [11].

For axisymmetric stress states the Matsuoka-Nakai model corresponds to the Mohr-Coulomb model. Nevertheless, the Matsuoka-Nakai model is described by a smooth surface in the stress space and thus it is more suitable from the computational aspect.

When the cohesion c is considered in the model, the yield condition is formulated for a modified stress [9]

$$\bar{\sigma}_{ij} = \sigma_{ij} - \sigma_0\delta_{ij}$$

with

$$\sigma_0 = c\cot\phi\ .$$

Notice that straightforward usage of the function f above also as flow rule g would lead to a loss of the information on ϕ in the derivative of g, since the ϕ only appears in a constant.

For that reason we apply as flow rule g the Drucker-Prager function (see elsewehere in this theoretical part). A separate ϕ_{flow} can be specified which enters this Drucker-Prager g. Since we use the Drucker-Prager function as flow rule g, the present plasticity model has been named *Modified* Matsuoka-Nakai.

<u>Mohr-Coulomb plasticity model</u>

The **group_materi_plasti_mohrcoul** model reads

$$0.5(\sigma_1 - \sigma_3) + 0.5(\sigma_1 + \sigma_3)\sin(\phi) - c\ \cos(\phi) = 0$$

Here c is the cohesion, σ_1 is the maximal principal stress and σ_3 is the minimal principal stress. The angle ϕ needs to be specified for both the yield condition and the flow rule; by choosing different values, non-associative plasticity is obtained.

For a numerically more stable solution, consider using Matsuoka-Nakai plasticity in stead of Mohr-Coulomb.

<u>Mohr-Coulomb softening plasticity model</u>

The **group_materi_plasti_mohrcoul_softening** model is the same as the standard Mohr-Coulomb model. Now, however, the parameters c and ϕ (both for the yield rule and for the flow rule) are softened on the the effective plastic strain κ.

By example, for the cohesion a linear variation is taken between the initial value c_0 at $\kappa = 0$, up to c_1 at a specified critical value of κ, and constant c_1 for larger values of κ. The same is done for ϕ for the yield rule and for the flow rule.

Isotropic Hardening

The size of the plastic strains rate is measured by the **materi_plasti_kappa** parameter

$$\dot{\kappa} = \sqrt{0.5\dot{\epsilon}_{ij}^{\text{plas}}\dot{\epsilon}_{ij}^{\text{plas}}}$$

This parameter can be used for isotropic hardening. Use the **dependency_diagram** for this.

Kinematic Hardening

The **materi_plasti_rho** matrix ρ_{ij}, governs the kinematic hardening in the plasticity models. It is used in the yield rule and flow rule to get a new origin by using the argument $\sigma_{ij} - \rho_{ij}$:

$$f^{\text{yield}} = f^{\text{yield}}(\sigma_{ij} - \rho_{ij})$$

$$f^{\text{flow}} = f^{\text{flow}}(\sigma_{ij} - \rho_{ij})$$

where the rate of the matrix ρ_{ij} is taken to be

$$\dot{\rho_{ij}} = a \;\; \dot{\epsilon_{ij}}^{\text{plas}}$$

where a is a user specified factor (see **group_materi_plasti_kinematic_hardening**).

Plastic heat generation

The plastic energy loss can be partially turned into heat rate per unit volume q:

$$q = \eta \, \sigma_{ij} \, \dot{\epsilon_{ij}}^{\text{plas}}$$

where η is a user specified parameter (between 0 and 1) specifying which part of the plastic energy loss is turned into heat (see **group_materi_plasti_heatgeneration**).

2.2.4 Hypo-Plasticity

In hypoplasticity a direct relation is used between strain rates and stress rates. Specifically:

$$\dot{\sigma}_{ij} = L_{ijkl}\dot{\epsilon}_{ij} + N_{ij}\sqrt{\dot{\epsilon}_{kl}\dot{\epsilon}_{kl}}$$

Here the part with L_{ijkl} gives a linear relation between strain rates and stress rates and the part with N_{ij} gives a nonlinear relation. The constitutive tensors L_{ijkl} and N_{ij} are functions of the effective stress tensor σ_{ij} and void ratio e. The effective stress tensor σ_{ij} follows from the total stress tensor σ_{ij} minus any pore pressures (see groundflow). Rigid body rotations (objectivity) are treated elsewhere (see the section on memory).

Basic law, Wolffersdorff

The law proposed by WOLFFERSDORFF [11] is used.

$$L_{ijkl} = f_s \frac{1}{\hat{\sigma}_{mn}\hat{\sigma}_{mn}} \left(F^2\, I_{ijkl} + a^2\, \hat{\sigma}_{ij}\hat{\sigma}_{kl}\right)$$

$$N_{ij} = f_s\, f_d \frac{F\,a}{\hat{\sigma}_{kl}\,\hat{\sigma}_{kl}} \left(\hat{\sigma}_{ij} + \hat{\sigma}^*_{ij}\right)$$

$$\text{with} \quad \hat{\sigma}_{ij} = \sigma_{ij}/(\sigma_{mn}\delta_{mn}) \quad , \quad \hat{\sigma}^*_{ij} = \hat{\sigma}_{ij} - \tfrac{1}{3}\,\delta_{ij} \quad , \quad I_{ijkl} = \delta_{ik}\delta_{jl} \ ,$$

$$a = \frac{\sqrt{3}(3 - \sin\varphi_c)}{2\sqrt{2}\sin\varphi_c}$$

$$F = \sqrt{\frac{1}{8}\tan^2\psi + \frac{2 - \tan^2\psi}{2 + \sqrt{2}\tan\psi\cos 3\theta}} - \frac{1}{2\sqrt{2}}\tan\psi \ ,$$

$$\tan\psi = \sqrt{3}\sqrt{\hat{\sigma}^*_{ij}\hat{\sigma}^*_{ij}} \quad , \quad \cos 3\theta = -\sqrt{6}\,\frac{\hat{\sigma}^*_{ij}\hat{\sigma}^*_{jk}\hat{\sigma}^*_{ki}}{[\hat{\sigma}^*_{mn}\hat{\sigma}^*_{mn}]^{3/2}} \quad .$$

For $\hat{\sigma}^*_{ij} = 0$ is $F = 1$.

The scalar factors f_s and f_d take into account the influence of mean pressure and density:

$$f_s = \frac{h_s}{n}\left(\frac{e_i}{e}\right)^{\beta}\frac{1+e_i}{e_i}\left(-\frac{\sigma_{ij}\delta_{ij}}{h_s}\right)^{1-n}\left[3 + a^2 - a\sqrt{3}\left(\frac{e_{i0} - e_{d0}}{e_{c0} - e_{d0}}\right)^{\alpha}\right]^{-1} \ ,$$

$$f_d = \left(\frac{e - e_d}{e_c - e_d}\right)^{\alpha} \quad .$$

Three characteristic void ratios – e_i (during isotropic compression at the minimum density), e_c (critical void ratio) and e_d (maximum density) – decrease with mean stress:

$$\frac{e_i}{e_{i0}} = \frac{e_c}{e_{c0}} = \frac{e_d}{e_{d0}} = \exp\left[-\left(-\frac{\sigma_{ij}\delta_{ij}}{h_s}\right)^n\right]$$

The range of admissible void ratios is limited by e_i and e_d. The model parameters can be found in Tab. 1. They correspond to Hochstetten sand from the vicinity of Karlsruhe, Germany [11].

φ [°]	h_s [MPa]	n	e_{c0}	e_{d0}	e_{i0}	α	β
33	1000	0.25	0.95	0.55	1.05	0.25	1.0

Table 1: *Basic hypoplastic parameters of Hochstetten sand.*

The basic law parameters should be specified in **group_materi_plasti_hypo_wolffersdorff**.

Cohesion

A simplistic appraoch to include cohesion is used here. Instead of feeding the real effective stress state σ_{ij} into the hypoplastic law, an alternative effective stress state σ^c_{ij} is used. Cohesion is modelled by subtracting in each of the normal stress components a value c representing cohesion: $\sigma^c_{11} = \sigma_{11} - c$, $\sigma^c_{22} = \sigma_{22} - c$ and $\sigma^c_{33} = \sigma_{33} - c$. The shear stresses are not altered: $\sigma^c_{12} = \sigma_{12}$, etc.

The cohesion value should be specified in **group_materi_plasti_hypo_cohesion**.

Intergranular strains

In order to take into account the recent deformation history, an additional tensorial state variable S_{ij}[1] is introduced.

Denoting the normalized magnitude of S_{ij}

$$\rho = \frac{\sqrt{S_{ij}S_{ij}}}{R}$$

(R is a material parameter) and the direction of S_{ij}

$$\hat{S}_{ij} = \frac{S_{ij}}{\sqrt{S_{kl}S_{kl}}}$$

($\hat{S}_{ij} = 0$ for $S_{ij} = 0$), the evolution equation for the intergranular strain tensor reads:

$$\dot{S}_{ij} = \begin{cases} (I_{ijkl} - \rho^{\beta_r}\hat{S}_{ij}\hat{S}_{kl})\dot{\epsilon}_{kl} & \text{for} \quad \hat{S}_{ij}\dot{\epsilon}_{ij} > 0 \\ \dot{\epsilon}_{ij} & \text{for} \quad \hat{S}_{ij}\dot{\epsilon}_{ij} \leq 0 \end{cases} ,$$

where $\dot{S}_{ij}$ is the objective rate of intergranular strain. Rigid body rotations are treated elsewhere (see the section on memory). From the evolution equation (2.2.4) it follows that ρ must remain between 0 and 1.

The general stress-strain relation is now written as

$$\dot{\sigma}_{ij} = M_{ijkl}\dot{\epsilon}_{kl} \quad .$$

The fourth order tensor M_{ijkl} represents the incremental stiffness and is calculated from the hypoplastic tensors L_{ijkl} and N_{ij} which may be modified by scalar multipliers m_T and m_R, depending on ρ and on the product $\hat{S}_{ij}\dot{\epsilon}_{ij}$:

$$\begin{aligned} M_{ijkl} &= [\rho^{\chi}m_T + (1-\rho^{\chi})m_R]L_{ijkl} + \\ &+ \begin{cases} \rho^{\chi}(1-m_T)L_{ijmn}\hat{S}_{mn}\hat{S}_{kl} + \rho^{\chi}N_{ij}\hat{S}_{kl} & \text{for} \quad \hat{S}_{ij}\dot{\epsilon}_{ij} > 0 \\ \rho^{\chi}(m_R - m_T)L_{ijmn}\hat{S}_{mn}\hat{S}_{kl} & \text{for} \quad \hat{S}_{ij}\dot{\epsilon}_{ij} \leq 0 \end{cases} \end{aligned}$$

χ is an additional material parameter.

An example intergranular parameters can be found in Tab. 2.

R	m_R	m_T	β_r	χ
$1 \cdot 10^{-4}$	5.0	2.0	0.50	6.0

Table 2: *Example of Intergranular hypoplastic parameters.*

The intergranular parameters should be specified in **group_materi_plasti_hypo_intergranularstrain**. Also you need to include **materi_strain_intergranular** in the initialisation part.

[1] S_{ij} is denoted δ_{ij} in the paper [10]. However, in order to avoid confusion with Kronecker delta, another symbol is used here.

Pressure dependent initial void ratio

You can correct the initial void ratio e_0, as specified in the initial value for the history variable in the **node_dof** records, for the initial pressure to obtain a corrected initial void ratio e.

$$\frac{e}{e_0} = \exp\left[-\left(-\frac{\sigma_{ij}\delta_{ij}}{h_s}\right)^n\right]$$

See the basic law description for the parameters h_s and n. The σ_{ij} denotes the effective stress tensor (total stresses minus any groundflow pressure). This pressure dependent initial void ratio correction can be activated by **group_materi_plasti_hypo_pressuredependentvoidratio**. After the initial void ratio has been established, the development of the void ratio is governed by volumetric compression or extension of the granular skeleton.

2.2.5 Damage

In the presence of **materi_damage** d, the **materi_stress** follows:

$$\sigma_{ij}^{\text{damaged}} = (1-d)\sigma_{ij}^{\text{undamaged}}$$

For the damage, the **group_materi_damage_mazars** model is available:

$$d = d_t\ \alpha^\beta\ +\ d_c\ (1-\alpha)^\beta$$

where

$$d_t = 1. - (1-a_t)\ \frac{\epsilon^0}{\epsilon^{\text{eq}}} - a_t\ e^{-b_t\,(\epsilon^{\text{eq}}-\epsilon^0)}$$

and

$$d_c = 1. - (1-a_c)\ \frac{\epsilon^0}{\epsilon^{\text{eq}}} - a_c\ e^{-b_t\,(\epsilon^{\text{eq}}-\epsilon^0)}$$

Here ϵ^{eq} contains the positive principal strains. The parameter α is given by the ratio $\frac{\epsilon^{\text{eq}}}{\epsilon}$, where ϵ contains the total strains (both negative and positive). The parameter ϵ^0 is the strain threshold for damage; other material parameters are β , a_t , b_t , a_c , b_c. Typically for concrete:

$1.e{-4} < \epsilon^0 < 3.e{-4}$; $\beta = 1.$; $1 < a_t < 1.5$; $500 < b_t < 2000$; $0.7 < a_c < 1.2$; $e^4 < b_c < 5e^4$

You can combine damage freely with plasticity models or other material behavior.

2.2.6 Average stress (hydrostatic compressibility)

An extra average stress contribution on each of σ_{11}, σ_{22} and σ_{33} is

$$\frac{1}{co}\frac{\partial v_i}{\partial x_i}$$

where co is the **group_materi_elasti_compressibility**, which should not be 0. This pressure term can e.g. be used to model nearly incompressible fluids. The compressibility contribution should be combined with a contribution for the deviatoric stresses (e.g. **group_materi_viscosity**).

2.2.7 Thermal stresses

Temperature rates cause fictitious thermal strain rates

$$-\alpha \dot{T} \delta_{ij} \quad \text{where} \quad \delta_{ij} = 1 \quad \text{if} \quad i = j \quad \text{else} \quad \delta_{ij} = 0$$

where α is the **group_materi_expansion_linear** coefficient and $\dot{T}$ is the **condif_temperature**. These fictitious thermal strain rates in turn lead to stress rates.

2.2.8 Hyper elasticity

Hyper elasticity is used to model rubbers. It should be combined with a total lagrange formulation for the memory of the material (so use **-total** or **-total_piola** for **group_materi_memory**).

The stresses follow from a strain energy function (with C_{ij} components of the matrix C, and where F is the deformation tensor and U is the stretch tensor following from the polar decomposition of the deformation tensor)

$$2\frac{\partial W}{\partial C_{ij}}$$

$$C = F^T F = U^T U$$

Deviatoric contributions

To obtain a purely deviatoric function, the following strain measures are used (with I_1, I_2 and I_3 the first, second and third invariant of the elastic strain matrix C respectively)

$$J_1 = I_1 I_3^{\frac{-1}{3}} \quad J_2 = I_2 I_3^{\frac{-2}{3}}$$

The **group_materi_hyper_besseling** function reads (with K_1, K_2 and α user defined constants)

$$W = K_1(J_1 - 3)^\alpha + K_2(J_2 - 3)$$

The **group_materi_hyper_mooney_blatz_ko** function reads (with G and β user defined constants)

$$W = G * 0.5 * (I_1 - 3.0 + (2.0/\beta)(J^{-\beta} - 1.));$$

This Blatz-Ko hyperelastic material hardens in compression, and softens slightly in tension; it models a foamlike rubber.

The **group_materi_hyper_mooney_rivlin** function reads (with K_1 and K_2 user defined constants)

$$W = K_1(J_1 - 3) + K_2(J_2 - 3)$$

The **group_materi_hyper_neohookean** function reads (with K_1 a user defined constant)

$$W = K_1(J_1 - 3)$$

The **group_materi_hyper_reducedpolynomial** function reads (with K_i user defined constants)

$$W = K_i(J_1 - 3)^i$$

where a summation over $i = 1, 2, \ldots$ is applied.

<u>Volumetric contributions</u>

We define $J = \sqrt{I_3}$. Now a volumetric part can be added to the strain energy.

The **group_materi_hyper_volumetric_linear** contribution reads:

$$W = \frac{K}{2}(J - 1)^2$$

The **group_materi_hyper_volumetric_murnaghan** contribution reads:

$$W = \frac{K}{\beta}(\frac{1}{\beta - 1}J^{-\beta} + 1)J$$

The **group_materi_hyper_volumetric_polynomial** contribution reads:

$$W = \frac{K_i}{2}(J - 1)^{2i}$$

for $i = 0, 1, \ldots$.

The **group_materi_hyper_volumetric_simotaylor** contribution reads:

$$W = \frac{K}{2}((J - 1)^2 + (lnJ)^2)$$

The **group_materi_hyper_volumetric_ogden** contribution reads:

$$W = \frac{K}{\beta}(\frac{1}{\beta}(J^{-\beta} - 1) + lnJ)$$

As an example, you can combine the **group_materi_hyper_mooney_rivlin** energy function with the **group_materi_hyper_volumetric_linear** so that the total strain energy function becomes:

$$W = K_1(J_1 - 3) + K_2(J_2 - 3) + \frac{K}{2}(J - 1)^2$$

Here the initial shear modulus and bulk modulus are included as:

$$\text{initial shear modulus} = 2(K_1 + K_2)$$

and

$$\text{initial bulk modulus} = K$$

respectively.

2.2.9 Viscoelasticity

Viscoelasticity is modeled with n parallel **group_materi_maxwell_chains**. Each of the chains contains a spring with stiffness E^m in line with a dash pot with relaxation time t^m (m indicates the m-th maxwell chain). The viscoelastic stress rate is given by (with C^m_{ijkl} is the elastic tensor modulus of the m-th maxwell chain (depending on E^m and the poisson ratio))

$$\sum_{m=0}^{m=n-1} ({C_{ijkl}}^m \dot{\epsilon_{kl}}^{\text{elas}} - \frac{{\sigma_{ij}}^m}{t^m})$$

2.2.10 Viscoplasticity

Viscoplasticity is a model for rate-dependent plasticity. Rate dependent plasticity is important for (high-speed) transient plasticity calculations. It should be used in combination with a plasticity law. Viscoplasticity influences the stresses via the plastic strains.

The **group_materi_plasti_visco_exponential** model reads

$$\dot{\epsilon_{kl}}^{\text{plas}} = \gamma \; p \; e^{\alpha f} \;\; \frac{\partial f^{\text{flow}}}{\partial \sigma_{kl}}$$

where γ and α are material fluidity constants and p is the pressure. This model was first developed for visco-plastic soil behaviour.

The **group_materi_plasti_visco_power** model reads

$$\dot{\epsilon_{kl}}^{\text{plas}} = \eta(\frac{f}{f^{\text{ref}}})^p \;\; \frac{\partial f^{\text{flow}}}{\partial \sigma_{kl}}$$

where η (fluidity constant), f^{ref} (reference value for yield function, e.g. the yield stress for Von-Mises plasticity) and p (power) are user specified parameters.

2.2.11 Viscosity

The viscous contribution to the total stress is

$$2\nu D_{ij}$$

where

$$D_{ij} = 0.5(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i})$$

and divergence is neglected since we only model slightly compressible flows.

Viscous heat generation

The viscous energy loss is turned into heat rate per unit volume q:

$$q = 2\nu D_{ij} D_{ij}$$

See **group_materi_viscosity_heatgeneration**.

2.3 Contact analysis

2.3.1 Penalty formulation

In contact analysis, normal forces F_n follow from the condition that bodies cannot penetrate each other. Since we use a penalty formulation, the normal force is given by

$$F_n = \lambda u_n$$

where u is the penetration and λ is called the **contact_penalty_velocity** because its generates forces on the velocity unknowns. You can also impose **groundflow_pressure** and **condif_temperature** contact conditions by respectively the penalty factors **contact_penalty_pressure** and **contact_penalty_temperature**.

2.3.2 Friction and frictional heat generation

This normal force leads to a friction force F_f which equals

$$F_f = \nu \, F_n$$

where ν is the *friction* coefficient (see **contact_friction**. The friction force causes heat generation rate Q:

$$Q = \eta \, F_f \, v_f$$

where v_f is the slip velocity, and the factor η is a user specified factor which determines which part of the frictional energy loss is transformed into heat (η is between 0 and 1; see **contact_heatgeneration**).

2.4 Ground water flow

2.4.1 Storage equation

The hydraulic pressure head h follows from the storage equation:

$$C\,\dot{h} = (k_1^p \frac{\partial^2 h}{\partial {x_1}^2} + k_2^p \frac{\partial^2 h}{\partial {x_2}^2} + k_3^p \frac{\partial^2 h}{\partial {x_3}^2}) + \frac{\partial v_i}{\partial x_i} + f$$

Primary unknown is the hydraulic pressure head **groundflow_pressure**. Further notation: C **group_groundflow_capacity**; k_i^p **group_groundflow_permeability** in i-direction; x_i space coordinate; v_i material velocity (if present); f **force_element_volume** (hydraulic pressure source). The equation is given for space coordinates following material velocities v_i (if present).

Groundflow velocities

The groundflow velocities, after initializing **groundflow_velocity**, follow from:

$$ {v_i}^{\mathrm{g}} = k_i^p \frac{\partial h}{\partial x_i}$$

Total groundwater pressure

The total groundwater pressure is by example needed to calculate the total stresses in soils (total soil stress = effective soil stress + total groundwater pressure). The total groundwater pressure follows from:

$$p_{\mathrm{total}} = h - \rho g z$$

where g is the gravitational acceleration, and ρ is the **groundflow_density** (Please notice that z typically is a negative number).

Static groundwater pressure

The static pressure due to gravity is:

$$p_{\mathrm{static}} = \rho g \Delta z$$

where the Δz is the distance to the groundwater level, the phreatic level. The phreatic level needs to be specified with the **groundflow_phreaticlevel** record. If that **groundflow_phreaticlevel** record is not specied, the static pressure part is not used, so that the static pressure becomes zero.

Dynamic groundwater pressure

The dynamic groundwater pressure follows from

$$p_{\mathrm{dynamic}} = p_{\mathrm{total}} - p_{\mathrm{static}}$$

Boundary conditions

If the groundwater velocity is 0 normal to an edge (say at the interface with a rock layer it is zero), then you should prescribe nothing on that edge (Tochnog will then take care of that boundary condition for you).

At the phreatic level where the groundflow meets free air the hydraulic pressure head should become $\rho g z$. You can either set this yourself by using **bounda_unknown** combined with **bounda_time** or else demand that Tochnog automatically does it for you by activating the option **groundflow_phreaticlevel_bounda**.

At edges where you have some other hydraulic head you need to specify that head yourself with **bounda_unknown** and **bounda_time** records.

If gravity is not of importance, e.g. in biomechanics where the storage equation is used to model fluid transport in soft tissues, the hydraulic pressure head h is equal to the total pressure, and thus is zero at edges where the water meets the free air. In this case, set h to zero by using **bounda_unknown** combined with **bounda_time**.

Postprocessing

For all printing, plotting etc. you normally get the hydraulic pressure head h since it is the primary unknown solved in the storage equation. The total pressure, static pressure and dynamic pressure you can get by the **post_calcul** option.

Naming conventions

To connect better to conventional naming, we rematk that the capacity depends on the porosity n and water compressibility β:

$$C = n\,\beta$$

and for the permeability:

$$k_i^p = \frac{k_i}{\rho\,g}$$

where k_i is the permeability in i-direction.

Consolidation analysis

Look in the 'Consolidation' section of the 'Interaction analyzes and advanced analyzes' chapter in the end of this manual on how to perform consolidation analyzes (combined groundwater flow with soil stress analyzes).

2.5 Wave equation

$$\frac{\partial \dot{s}}{\partial t} = c^2 \left(\frac{\partial^2 s}{\partial {x_1}^2} + \frac{\partial^2 s}{\partial {x_2}^2} + \frac{\partial^2 s}{\partial {x_3}^2} \right)$$

$$\frac{\partial s}{\partial t} = \dot{s}$$

The primary unknowns are the **wave_scalar** s and its first time derivative **wave_fscalar** $\dot{s}$ (as TOCHNOG only solves first order in time equations, the first time derivative of s also becomes primary unknown in order to turn this second order in time equation into a set of first order in time equations). Further notation: x space coordinate, t time and c speed of sound.

3 Examples

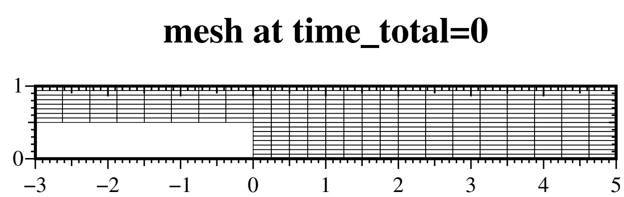
Data files are enclosed with the TOCHNOG distribution. These files have several purposes. First, they show the analysis capabilities of TOCHNOG. Second, these input files enable an easy start in performing calculations yourself. Third, the data items **target_item** and **target_value** are used in the data part, allowing you to test TOCHNOG on your specific computer system. This manual only contains results with some of the input files (examp1.dat, examp2.dat, etc..).

3.1 Example 1: Backward facing step (fluid flow)

The following specific case of the GAMM workshop [6] is studied here:

total length of channel $L = 22.$
length inflow section $l = 3.$
total height $H = 1.$
height inflow section $h = 0.5.$
viscosity $\nu = 0.01.$
density $\rho = 1.$
maximum flow component of parabolic profile in inflow section $U_{max} = 1.$
the flow volume is $(2/3)hU_{max} = 1/3.$
Reynolds $Re = U_{max}(H - h)/\nu = 50.$

The result for the minimum x-velocity v_x at distance $x = 0.8$ behind the step is studied. This result checks the correct modeling of the recirculation zone after the step. At the distance $x = 0.8$ we find the minimum velocity is $v_x = -0.041$. This happens at $y = 0.1$ above the lower edge. The results of the GAMM workshop were in between -0.066 and -0.018.



3.2 Example 2: Confined compression of fluid filled porous material

In this 1D example, a porous solid saturated with fluid is compressed by a force F on the solid phase. The length of the test specimen is L, plane strain conditions hold, the Young's modulus is E, Poisson's ratio is ν, permeability k_p, width of test specimen W, thickness T, area $A = WT$. Some non-dimensional variables are introduced

$$X = x/L \qquad T = \frac{Hk_p}{L^2}t \qquad P = \frac{Ap}{F}$$

$$H = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}$$

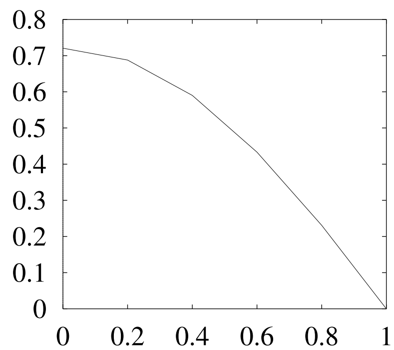
At $X = 1$ free drainage holds ($P = 0$) for $T > 0$. At $X = 0$ the boundary is not permeable ($\frac{\partial P}{\partial X} = 0$) for $T > 0$. Using the non-dimensional variables, the analytical solution reads (see [1])

$$P = \sum_{n=0}^{\infty} \frac{2}{M} \sin(MX) \exp(-M^2 T)$$

$$M = \frac{\pi}{2}(2n+1)$$

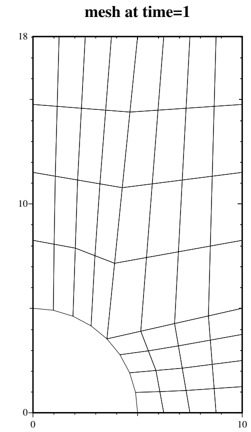
As a typical result, P at $T = 0.22$ is checked; at $X = 0.4$ the exact solution for the non-dimensional pressure P is 0.6.

Now we choose: $L = 1$, $W = 1$, $T = 1$, $E = 1.e1$, $\nu = 0$, $k_p = 0.1$ and $F = 1$. Thus, for $x = 0.4$ at $t = 0.22$ we should find $p = 0.6$. The numerical analysis (5 elements, time step size 0.01) gives $p = 0.607$. The pressure distribution at $T = 0.22$ is given below



3.3 Example 3: Plasticity in plate with circular hole

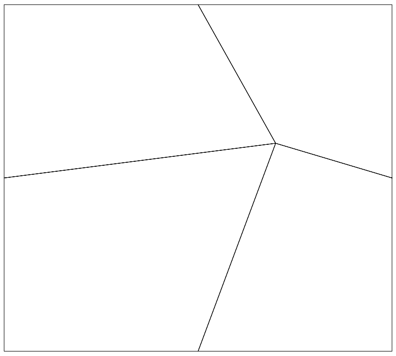
This example is taken from [13] (page 245). A plate (20 mm wide, 36 mm long) is perforated at the center with a hole with radius 5 mm. The plate is stretched in length direction by increasing the overall strain ϵ in time. The other two boundaries are free to deform. Because of symmetry, only a quarter needs to be modeled. A mesh with 48 linear quadrilateral elements is used



The Young's modulus $E = 7000\ \frac{kg}{mm^2}$, Poisson's ratio $\nu = 0.2$, the Von Mises yield stress $\sigma_y = 24.3\ \frac{kg}{mm^2}$ and the strain hardening $h = 0.032E = 224\ \frac{kg}{mm^2}$. Plane stress conditions hold. The plasticity influence is characterized by σ^{mean} at the upper edge for $v = 0.0625$. The elasto-plastic analysis gives $\sigma^{\text{mean}} = 12.4\ \frac{kg}{mm^2}$ for $v = 0.0625$. A purely elastic calculation would have given $\sigma^{\text{mean}} = 16.13\ \frac{kg}{mm^2}$.

3.4 Example 4: Two-dimensional convection and diffusion

This example demonstrates the effect of the spatial stabilization algorithm in 2D. A convection and diffusion of heat equation is analyzed on a 1 by 1 square. The two-dimensional mesh consists of distorted linear quadrilaterals



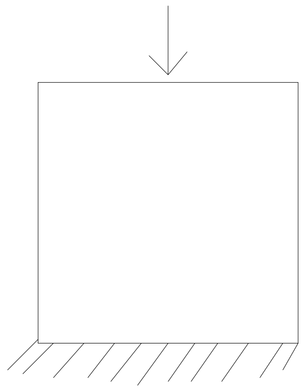
The convection velocity $\beta = 1$ and the conductivity $k = 0.01$. The boundary conditions for temperature are chosen such that the exact solution for a boundary layer in y-direction holds:

$$T(y) = T(y=0) + (T(y=1) - T(y=0))\frac{1 - \exp(\beta \frac{y}{k})}{1 - exp(\frac{\beta}{k})}$$

where we choose $T(y=0) = 1$ and $T(y=1) = 0$. This is a severe test for the spatial stabilization algorithm. Many algorithms exist which solve this example exactly when using a one-dimensional domain, say with y-axis only, but few exist which do not show wiggles for irregular 2D grids. The **node_dof** records are initialized with temperature 1 as a first estimate for the solution field. The we check the results at $x = 0.7$ and $y = 0.6$. The exact solution is 1. The numerical solution with the 4-noded elements is 0.95. Splitting the elements in triangles (see **control_mesh_split**) would have given the solution 1.001. Triangles seem to behave better than distorted quads (in this example anyway). Both solutions are quite good however.

3.5 Example 5: Inverse modeling of indenter test

In an experiment, a plate (size 1 by 1) is clamped at the bottom and loaded at the middle top point by a concentrated load ($-0.000\,1$ in y-direction)

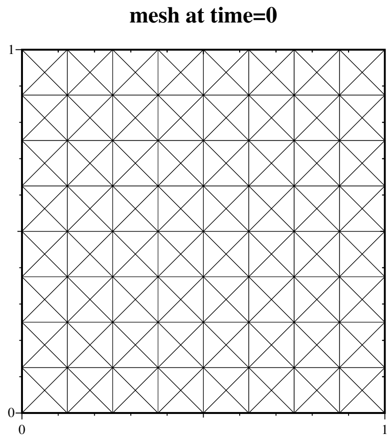


The y-displacement at the point of loading is measured ($-0.000\,25571$) and used to estimate the Young's modulus of the material. Four quadratic elements are used to model the indenter test. Initially, the Young's modulus is estimated to be 0.2. The top force and the measured displacement are such that the Young's modulus should become 1 in the inverse iteration process (with this element mesh). The table below shows the estimated Young's modulus versus the inverse iteration number

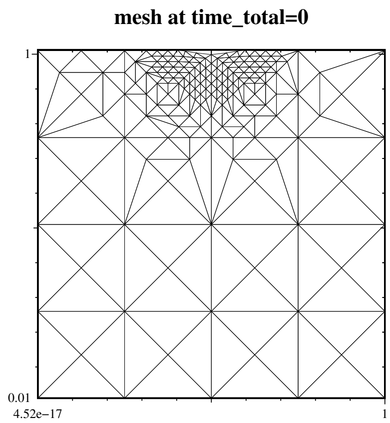
iteration	Young's modulus
0	0.200
1	0.276
2	0.371
3	0.487
4	0.619
5	0.760
6	0.887
7	0.970
8	0.998
9	1.000

3.6 Example 6: Impact of metal plate, local mesh refinement

A plate has an initial velocity and impacts on a rigid disk. The initial mesh is plotted below

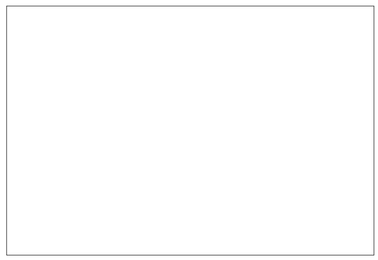


With this initial mesh, several time steps are taken. Then a local mesh refinement is applied. This local mesh refinement uses the residue in the equations to decide where the mesh should be refined. This strategy is applied several times. The final mesh is plotted below:

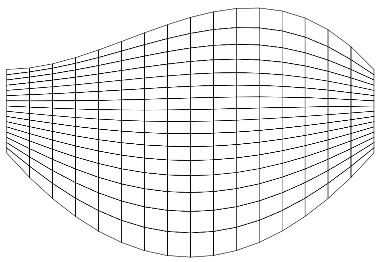


3.7 Example 7: Global mesh refinement over polynomial domain

This example shows that mesh refinement follows the boundaries of a domain; more precisely, if the nodes of an element edge are all placed on a specific geometrical entity (**geometry_line**, **geometry_circle**, etc.) then new generated nodes along that edge will also be placed on that geometrical entity; see **refine_globally_geometry** for this. Here we mesh a domain bounded at the left and right side by vertical lines ($x = 0$ and $x = 10$), bounded at the bottom by the polynomial $y = -10 - 10x + x^2$ and bounded at the top by the polynomial $y = +10 + x^2 - 0.1x^3$. Initially only one 4-noded quadrilateral is used



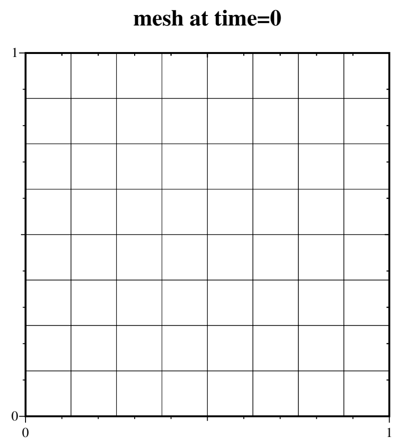
The four nodes of the element are placed on the intersection points of the polynomials and the lines. After 4 global refinements the mesh looks like



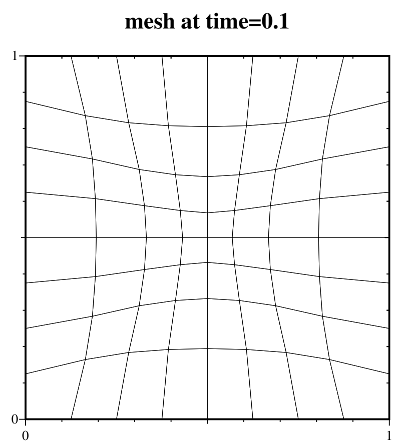
Each refinement is accompanied by some remeshing in order to obtain a more regular mesh.

3.8 Example 8: Residual remeshing near heat spot

The boundaries of a rectangle (size 1 by 1) have prescribed temperature 0.



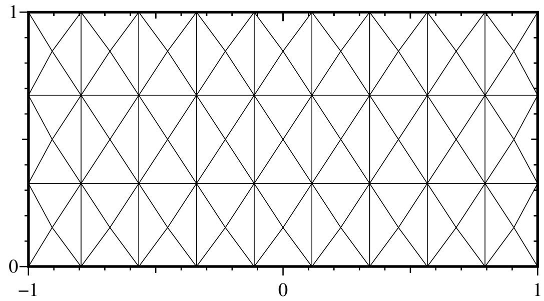
The middle point is heated so that it gets temperature 1. The residual remeshing is used to capture the temperature distribution near the heated spot in the middle. After some time, the temperature in the middle is imposed, and the residual remeshing did locate nodes more closely to the middle. The number of nodes is not changed; the nodes are replaced in the direction of high equation residual.



3.9 Example 9: 2D Impact of metal plate, rebuilding of mesh after large deformations.

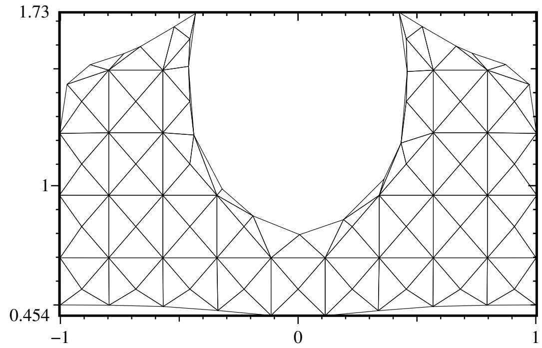
A metal plate impacts on a rigid sphere. We study here the large deformation of the plate. The initial mesh is show below

mesh at time_total=0



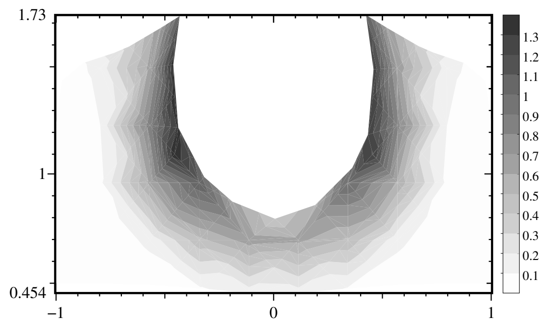
In each time step, a new mesh is built to prevent the elements from becoming too distorted. State variables (velocities, stresses, etc.) are mapped from the old mesh onto the new mesh. The mesh after these time steps is shown below:

mesh at time_total=0.0005



Finally we show the equivalent plastic strain distribution:

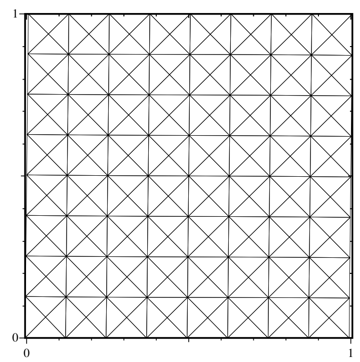
sepp at time_total=0.0005



3.10 Example 10: Shear of low-tension material (cracking)

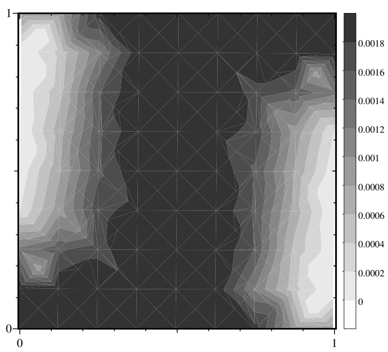
The top edge of a 1 by 1 plate is sheared over 0.01 to the right, and the bottom edge of the plate is sheared over a distance 0.01 to the left. The material cannot sustain positive principal stress exceeding 0.002 — it cracks. This cracking is modeled with a plasticity model which does not allow for principal stresses exceeding 0.002; the **group_materi_plasti_tension** model is used for this. The plate is modeled with linear triangles (**-tria3**). The first plot below shows the (deformed) mesh.

mesh at time=0.5



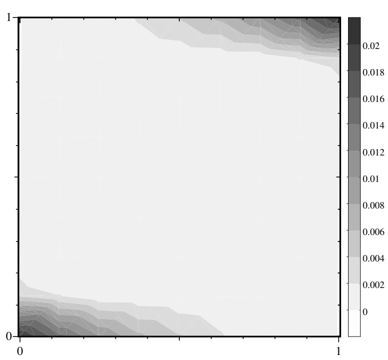
The second plot below shows the largest positive principal stress; notice that this principal stress did become equal to 0.002 over the entire tension diagonal.

stress−0 at time=0.5



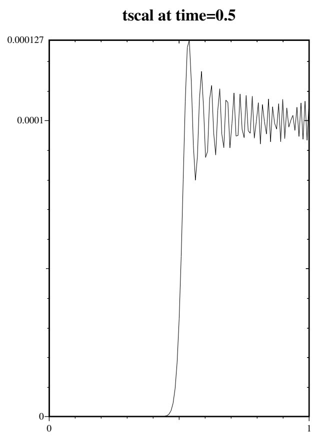
The third plot below shows the plastic strain; plastic strain does show up especially in the corners of the tension diagonal.

strain_plasti−size at time=0.5

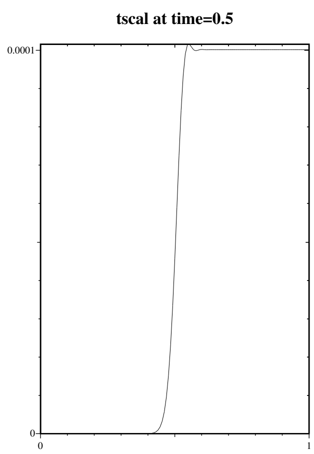


3.11 Example 11: Propagation of a disturbance in the wave equation.

The wave equation is used over a one-dimensional domain (between $x = 0$ and $x = 1$). The domain is divided into 128 linear elements. At time $t = 0$, $s = 0$ over the entire domain. After time $t = 0$, at $x = 0$ the scalar s is prescribed to hold the value 0 and at $x = 1$ the rate of the scalar s ($\dot{s}$) is prescribed to have the value $1.\ 10^{-4}$. This disturbance at the right edge propagates into the domain with the speed of sound ($c = 1$). At time 0.5 the rate of s over the entire domain is monitored; at this time point $\dot{s}$ should have become $1.\ 10^{-4}$ in the right half of the domain, whereas nothing should have happened yet in the left half of the domain. The first plot shows $\dot{s}$ if we use purely explicit time stepping (**control_timestep_iterations** is set to 1).

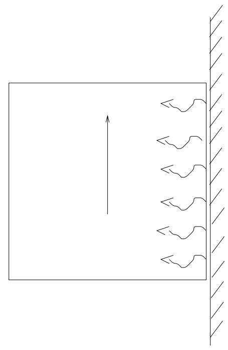


It is clear that the disturbance did propagate into half of the domain, but quite some oscillations do show up. The oscillations are greatly reduced if two iterations are used (**control_timestep_iterations** is set to 2); see the second plot.

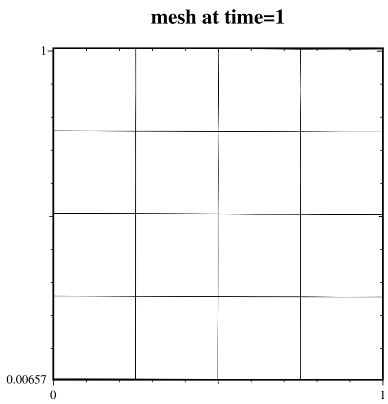


3.12 Example 12: Contact frictional heat generation.

A rectangular piece of material (size 1 by 1; Young's modulus 1) is rubbed against a solid wall. In this example, we will analyze the stresses and temperature profile caused by frictional heat generation. The free edges of the material are prescribed to have temperature 0.



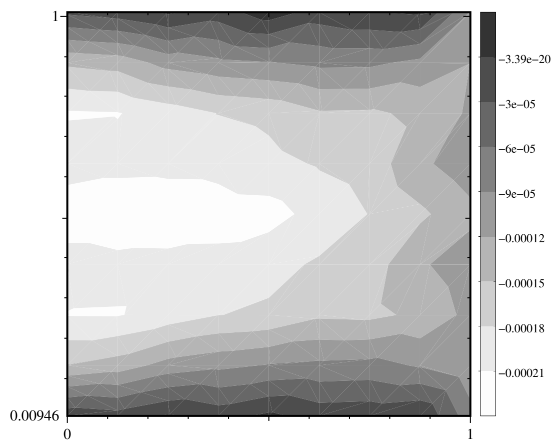
The deformed mesh (with 16 quadratic elements) at time 1 looks like:



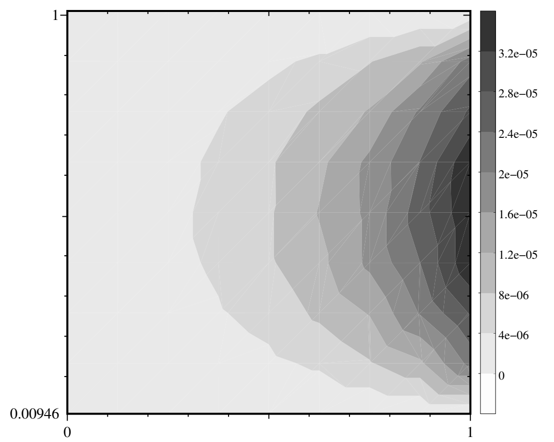
Initially the right edge of the material did penetrate the solid wall over a distance 0.01, but the contact algorithm did eliminate this penetration. This resulted in a normal stress **-sigxx** of about size −0.01. The normal stress causes a frictional stress **-sigxy** of about size −0.0001 at the interface between material and solid wall, due to a friction coefficient of 0.01:

This frictional stress, in turn, causes frictional heat generation. The stationary temperature profile is plotted below:

sigxy at time_current=10

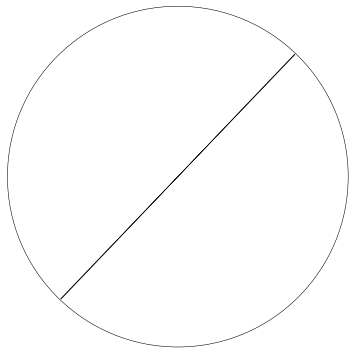


temp at time_current=10



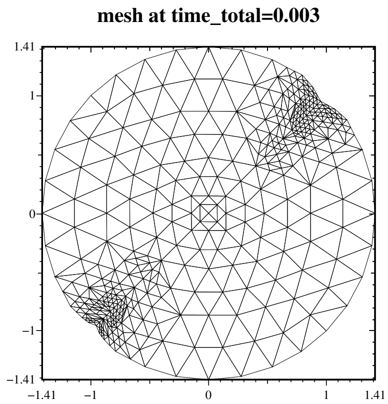
3.13 Example 13: Automatically embedded tendon with local mesh refinement

A tendon is embedded over a diagonal of a circular solid.

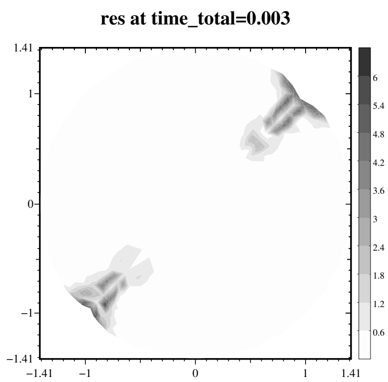


This tendon has an initial stress. Due to this initial stress, the tendon will contract and deform the solid. The solid is modeled with linear triangular elements. TOCHNOG will automatically distribute the tendon over the solid elements (automatic embedment).

Time steps are taken, after which the mesh is automatically locally refined; this is done several times. The first plot shows the final deformed mesh. Notice that most refinement takes place where the tendon enters into the solid, but there is also some refinement of elements along the tendon.



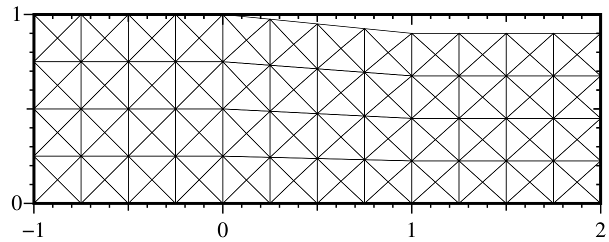
The second plot shows the residue in the equations. Notice that the residue is high at the entrance of the tendon in the solid, but high residues also appear on both sides of the tendon.



3.14 Example 14: Continuous metal forming.

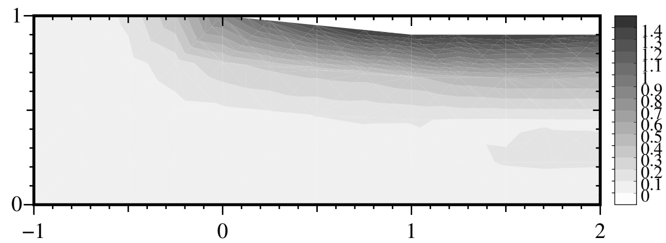
A metal sheet is extruded in a continuous process. The initial thickness of the metal is 1, and the thickness after extrusion is 0.9. The Young's modulus is 7000 and the Poisson ratio is 0.2. The Von Mises plasticity model is used to model permanent deformations (yield stress 24.3). The first flow shows the finite elements mesh:

mesh at time_current=0



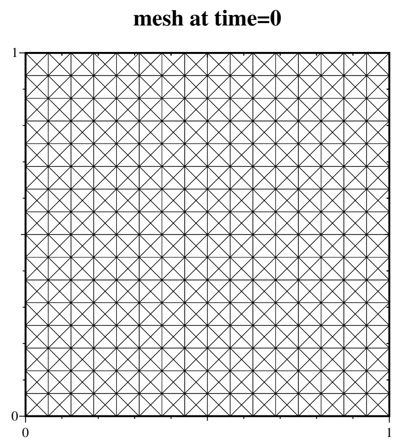
The extrusion is modeled by prescribing a velocity 1.0 at the right edge of the computational domain. On the left edge, the stresses and plastic strains are set to zero, in order to specify that virgin material enters the domain. Friction at the upper wall is not taking into account in this example. The result for κ, measuring the build up of plastic strains, is plotted below

kap at time_current=5

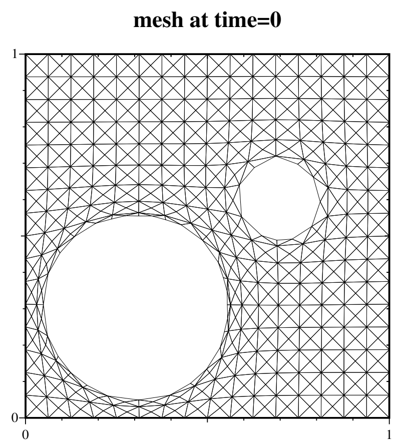


3.15 Example 15: Generation of two holes in 2D mesh.

This example demonstrates some mesh manipulation. We start with one 4-noded quadrilateral element (of size 1 by 1). After four global refinements of the mesh, and splitting the 4-noded elements into 3-noded triangles the following mesh is obtained:



Finally a two circular holes are taken out of the mesh, and 10 remeshing passes are applied to get more nicely shaped elements



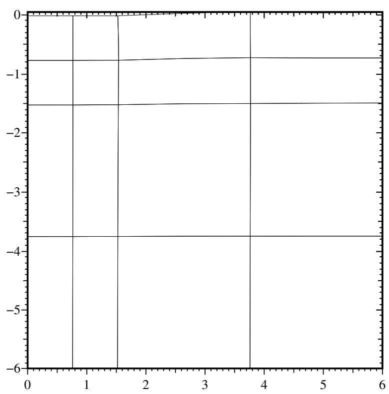
3.16 Example 16: Bearing capacity of foundation.

In this plane-strain example the Mohr-Coulomb plasticity law is used to calculate the bearing capacity of a foundation. The foundation is symmetric, so that only half of the problem is analyzed. The width of (half of) the foundation is 1.52m. The size of the soil domain is taken to be 6m by 6m. The Young's modulus of the soil is $207 \times 10^6 \mathrm{Nm}^{-2}$ and the Poisson ratio is 0.3. For the Mohr-Coulomb law, both the yield rule angle and the flow rule angle are 20 degrees, and the cohesion is $0.069\ 10^6 \mathrm{Nm}^{-2}$. At the bottom of the domain, all displacements are assumed to be fixed. At the left edge (the symmetry axis) and at the right edge, the horizontal displacement is fixed while the vertical displacement is free.

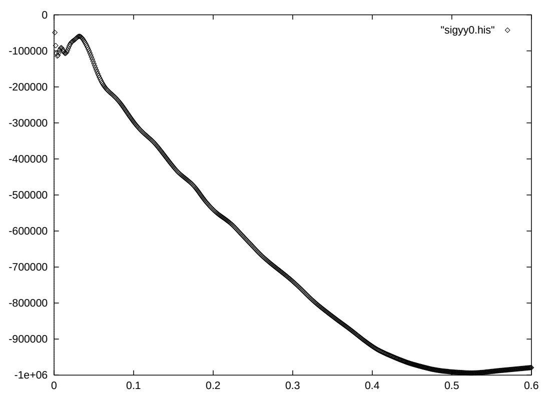
The classical solutions from Prandtl, Coulomb and Terzaghi give for the maximal average pressure σ_{yy} at the foundation values in the range $0.98\ 10^6 \mathrm{Nm}^{-2}$ up to $1.20\ 10^6 \mathrm{Nm}^{-2}$.

To get the same order of accuracy as in the classical solutions, a coarse mesh with only sixteen quadratic elements is used. For optimal numerical stability, the elements are integrated fully (integration points in the nodes). The foundation is prescribed a downward velocity of $0.05 \frac{\mathrm{m}}{\mathrm{s}}$ in the calculation. With time steps of 10^{-3}s the solution is advanced up to time 0.6s. First the deformed mesh at time 1s is plotted below.

mesh at time=0.6



Secondly, the development for the average pressure at footing over time is plotted

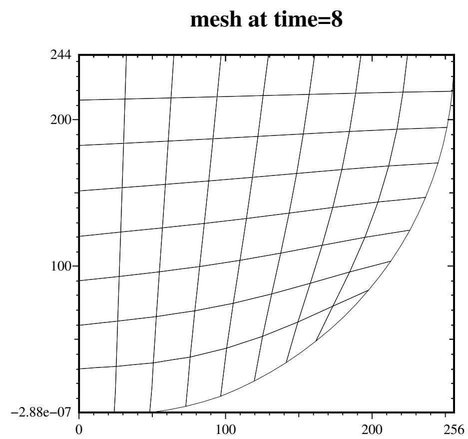


The maximum value of average pressure is near $1.1\ 10^6 \mathrm{Nm}^{-2}$.

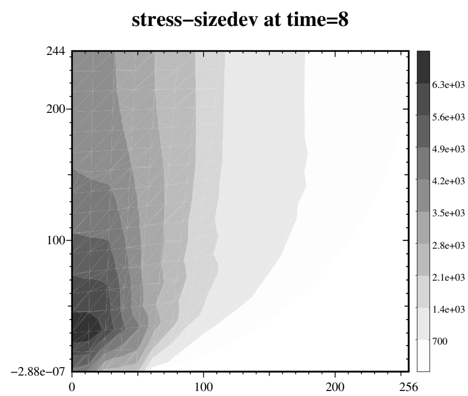
3.17 Example 17: Hertz contact problem.

A circular solid is compressed to a rigid surface. Due to symmetry only half of the circle needs to be analyzed. The compression is imposed, by prescribing the displacement of the middle line of the circle, so that in fact only one quarter of the circle needs to be analyzed. Plane strain conditions are assumed. The circle radius is 254 mm, the prescribed displacement of the middle line is 10.16 mm, the Young's modulus is 206000 Nmm^{-2} and the Poisson ratio is 0.3. The mesh contains 64 quadratic **-quad9** elements.

The first plot shows the deformed mesh. Notice that the nodes at the bottom of the circle did not penetrate the line $y = 0$.

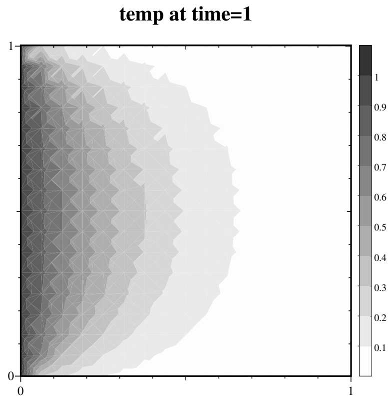


The second plot shows the Mises stresses (the size of the deviatoric stress matrix) in Nmm^{-2}..

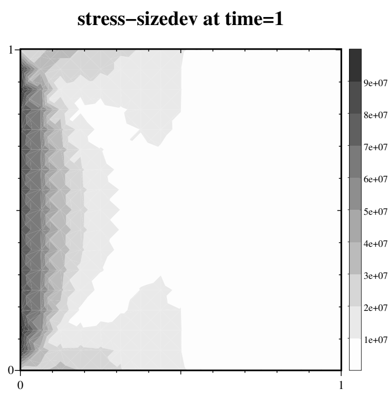


3.18 Example 18: Thermally induced stresses in plate.

A plate (of size 1 by 1) is initially stress free and in thermal equilibrium with its environment (all initial temperatures are 0). Then the left edge of the plate is prescribed a temperature 1 which will heat the plate. The plate starts convecting energy at its other edges; the coefficient for heat convection is 10^3, the convection environmental temperature is 0 and the conductivity of the plate is 50. The first plot shows the stationary temperature distribution. A mesh with 16 by 16 linear finite elements is used for the numerical analysis.



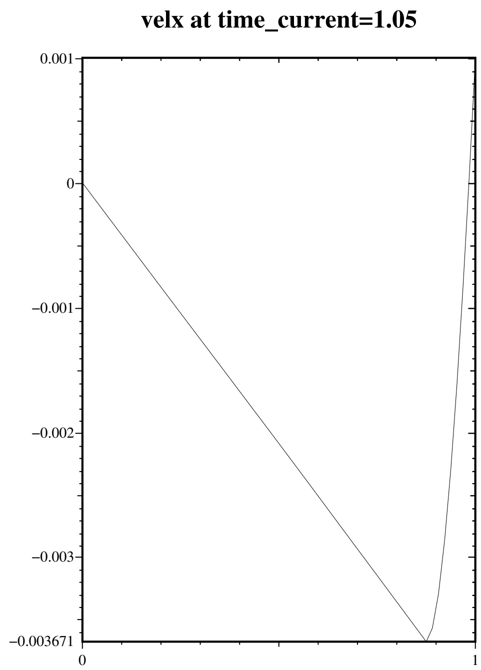
Due to the temperatures, thermal stresses will be induced in the plate. The plate cannot deform at its left edge, but is otherwise free to deform. The Young's modulus is 206 10^9, the Poisson ratio is 0.3, and a plane stress situation is assumed (σ_{zz} is 0). The thermal expansion coefficient is 8.4 10^{-4}. The second plot shows the size of the deviatoric stresses (the von Mises stress) which indicates if plasticity is to be expected.

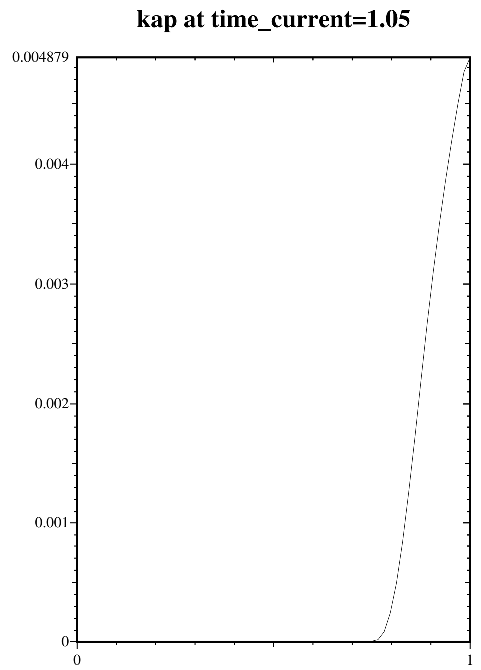


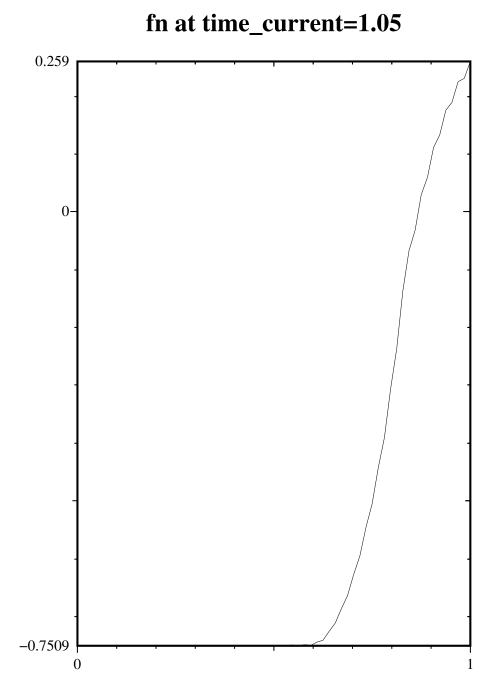
3.19 Example 19: Nonlocal plasticity in softening bar.

A bar is clamped at the left edge and is subjected at the right edge to a prescribed deformation. A plasticity model is used to limit the possible tensile stress in the bar. This limit of the tensile stress is softened to zero with the effective plastic strain κ. Due to softening, unlimited localization would be possible. This is prevented, however, by applying a viscoplastic power law in combination with a nonlocal yield rule, see **options_nonlocal**. A nonlocal radius of 0.2 is used in the determination of the nonlocal yield rule.

At the right side of the bar we introduce a relatively weak spot, so that plasticity occurs first there. In the plots below the results for the velocity field v_x, the effective plastic strain κ and the nonlocal yield rule fn respectively are shown. Please notice that the softening zone is of a limited, non mesh dependent, size.



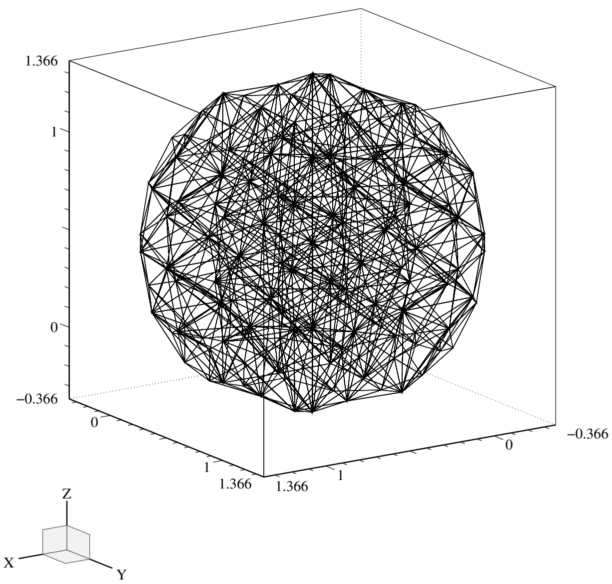

kap at time_current=1.05
0.004879
0.004
0.003
0.002
0.001
0
0
1


fn at time_current=1.05
0.259
0
−0.7509
0
1

3.20 Example 20: Three dimensional local mesh refinement near heat source.

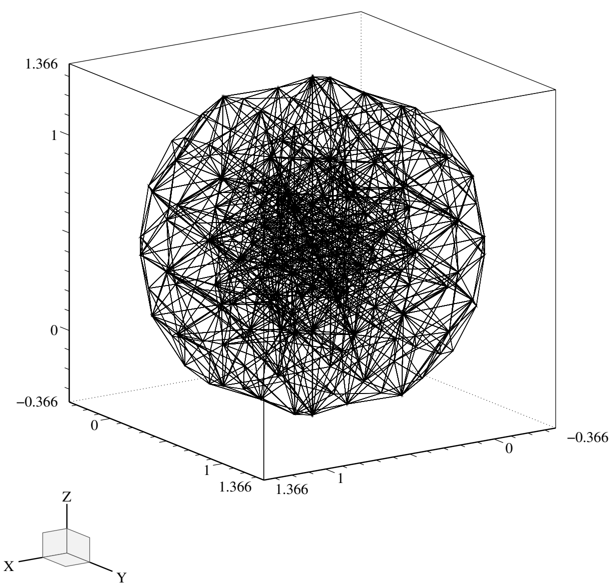
A sphere has zero temperature at the outer surface, and contains a heat source in the middle such that the temperature in the middle is 1. The temperature distribution is calculated. A wire frame plot of the initial mesh (with **-TET4** elements) is given below

mesh at time=0



Using the local refinement algorithm, 10 percent of the elements are refined after the time stepping shown in the figure. The **residue** in the temperature equation is used to select which elements need to be refined. Since the **residue** in the temperature equation is most large near the heat source in the middle, the local refinement algorithm places most elements near the middle. This can be seen in the wire frame plot below

mesh at time=1



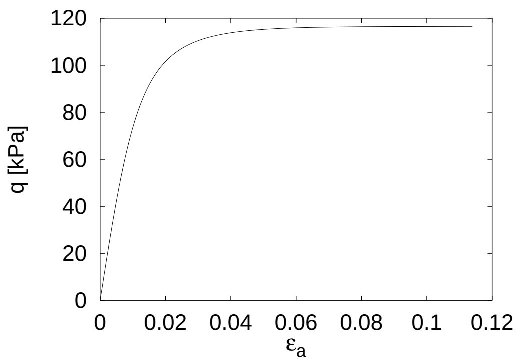
3.21 Example 22: Constant volume cylindrical compression with Modified Cam Clay.

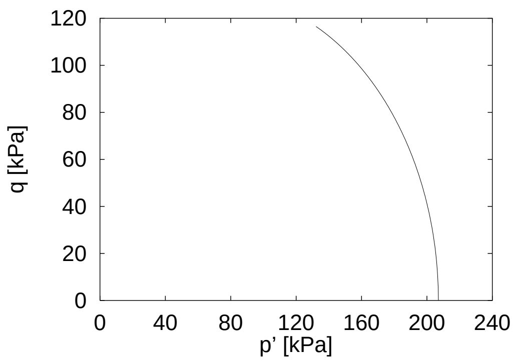
In soil mechanics, a constant volume cylindrical compression represents the so-called undrained triaxial test. The calculation was performed using a single -quad4 element under axisymmetric conditions. Only small deformations were considered. The parameters of the Modified Cam Clay model correspond to the Weald clay according to [2]: $M = 0.882$, $\lambda = 0.088$, $\kappa = 0.031$, $G = 3000$ kN/m^2. Initial (effective) stress was isotropic with the mean stress $p' = 207$ kN/m^2. The void ratio $e = 0.627$ remained constant during the compression due to the imposed constant volume condition. The initial preconsolidation mean pressure was $p_0 = 207$ kN/m^2.

The calculated stress-strain curve (q is the stress deviator and ε_a is the compressive axial deformation) in the left figure corresponds exactly to the numerically integrated equations of the model. The stress path in the right figure coincides with the analytical solution, see [12]:

$$\frac{p'_i}{p'} = \left(\frac{M^2 + \eta^2}{M^2 + \eta_i^2}\right)^{\Lambda}$$

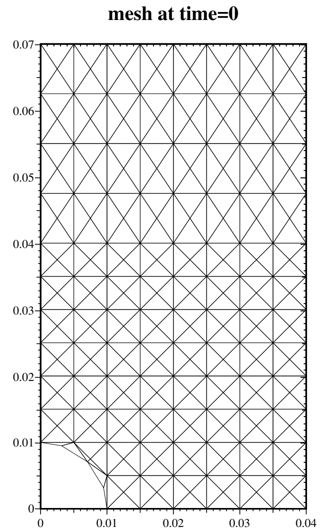
where $\eta = q/p'$ and $\Lambda = (\lambda - \kappa)/\lambda$. The subscript i denotes initial values.



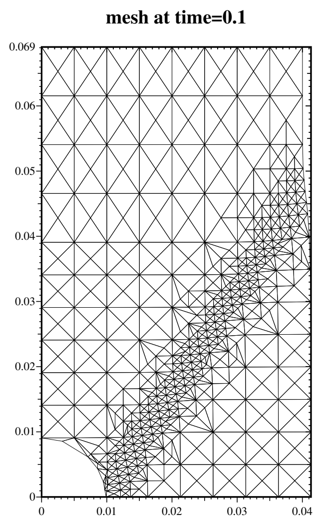


3.22 Example 23: Automatic local mesh refinement near shear band.

A test specimen is subjected to a vertical compression. The initial mesh is given in the first plot

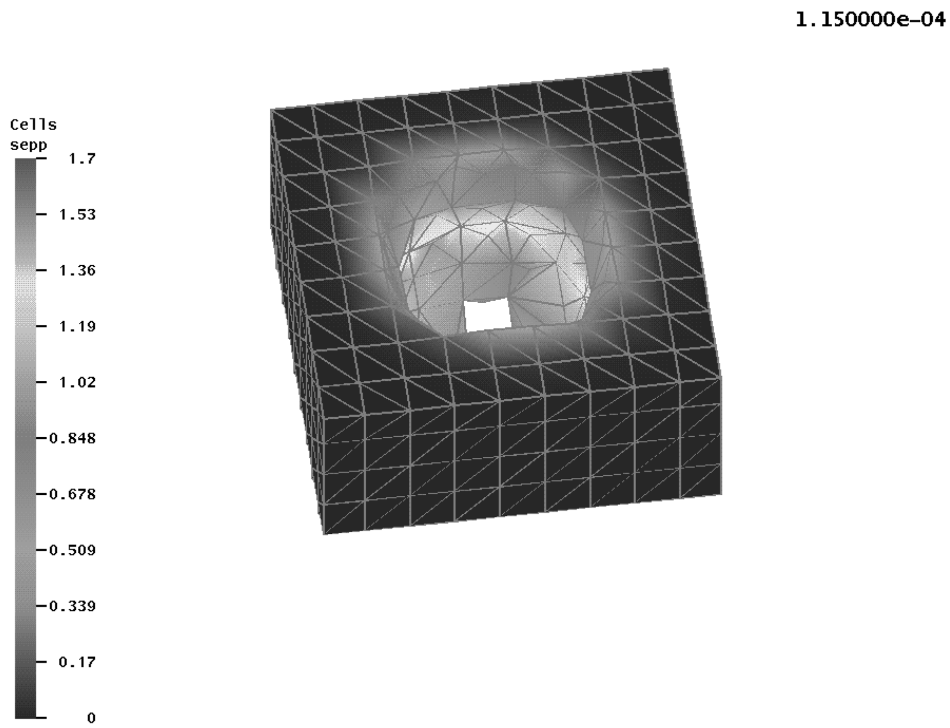


A hole is present in the down left corner; this hole triggers a shear band. Symmetry boundary conditions are applied, so that in fact the quarter part of a total test specimen is modeled. After the test specimen is compressed, automatic local mesh refinement is applied; the elements with the highest plastic strains are refined. Then the calculation restarts with the new mesh automatically. This refinement + automatic restart is done three times. The final plot shows the deformed mesh after the three refinements.



3.23 Example 24: 3D Impact of metal plate, rebuilding of mesh after large deformations.

This is essentially the same calculation as example 9, but now in 3D however. The metal plate is modeled with **-tet4** elements. At the end time, the sphere starts completely penetrating the plate. We only show the equivalent plastic strain distribution in the final mesh.



4 Input file, general remarks

The input is free format. Comments are enclosed between ( ), e.g. (this is comment only). The input should consist of an initialization part and a data part, separated by **end_initia** and ended by **end_data**

initialization
...
initialization
end_initia
data_item index data_values
...
data_item index data_values
end_data

Bold printed data in this manual can be used literally. *Italic* printed data is only symbolic (it represents a number or a word).

5 Input file, initialization part

The initialization part contains initialization records and an **end_initia** record

initialization
...
initialization
end_initia

The example below initializes a solid material

echo -yes
number_of_space_dimensions 2
materi_velocity
materi_strain_total
materi_stress
end_initia

The **echo** (always the first record), **number_of_space_dimensions** (always the second record), and **end_initia** record should be supplied always. Use **echo -yes** to echo the input and **echo -no** to not echo the input. Use **number_of_space_dimensions 1** for 1D problems, etc.. The records **materi_velocity**, **materi_strain_total** and **materi_stress** create a velocity, strain and stress field in the entire domain. In the following sections, all possible *initialization* records are discussed. Most of these records create an unknown field, a physical field like a temperature field or or a strain field, over the computational domain.

5.1 echo *switch* (first record of initialization part)

If *switch* is **-yes** the input will be echoed. If *switch* is **-no** the input will not be echoed. This needs to be the first record.

5.2 number_of_space_dimensions *number_of_space_dimensions* (second record of initialization part)

Set *number_of_space_dimensions* to 1 in 1D, etc.. You can also use the short name **ndim** if you prefer. This needs to be the second record.

5.3 derivatives (third record of initialization part, if specified)

If this record is included, the time derivative and the space derivatives will be stored in the **node_dof** records. This is only required for a limited number of models. The model description will specify if this **derivatives** initialization is needed.

5.4 condif_temperature

The temperature T is added to the **node_dof** records.

5.5 groundflow_velocity

The ground water flow velocity $v_i{}^g$ is added to the **node_dof** records. Condition: the **-groundflow** differential **group_type** should be present.

5.6 groundflow_pressure

The pressure p is added to the **node_dof** records. Condition: the **-materi** or **-groundflow** differential **group_type** should be present.

5.7 materi_damage

The damage d is added to the **node_dof** records. Also **materi_velocity** and **materi_strain_total** should be initialized.

5.8 materi_density

The density ρ is added to the **node_dof** records. Now this density is used in the equations, and the data item record **materi_density** in the data part needs not to be specified. The density is governed by the mass conservation law

$$\dot{\rho} + \frac{\partial \rho v_j}{\partial x_j} = 0$$

where v_j is the velocity in the direction of the space coordinate x_j. Don't forget to initialize the density by specifying **node_dof** records. Condition: the **-materi** differential **group_type** should be present.

5.9 materi_displacement

The displacements u, v, w are added to the **node_dof** records. If **materi_displacement** is initialized, then equations like the convection and diffusion of heat equation or the ground water flow equation are evaluated over the deformed volume, which is the sum of the nodal coordinates plus its displacements. Also if **materi_displacement** is initialized, the total lagrange model will be used in stress analysis.

Condition: also **materi_velocity** should be initialized because the displacement follows from integration of the velocity.

5.10 materi_history_variables *number_of_variables*

Generic history variables which can e.g. be used in some user supplied routines.

5.11 materi_maxwell_stress *number_of_chains*

Maxwell stress $\sigma_{11}{}^m$ $\sigma_{12}{}^m$ $\sigma_{13}{}^m$ $\sigma_{22}{}^m$ $\sigma_{23}{}^m$ $\sigma_{33}{}^m$ is added to the **node_dof** records. The parameter *number_of_chains* should match data item **group_materi_maxwell_chains**. The number of maxwell stresses is 6 * *number_of_chains*. Condition: the **-materi** differential **group_type** should be present.

5.12 materi_plasti_f

The plastic yield rule f is added to the **node_dof** records.

5.13 materi_plasti_f_nonlocal

The nonlocal plastic yield rule fn is added to the **node_dof** records. See also: **options_nonlocal**.

5.14 materi_plasti_kappa

The plastic isotropic hardening parameter κ is added to the **node_dof** records. Condition: the **-materi** differential **group_type** should be present.

5.15 materi_plasti_rho

The plastic kinematic hardening vector ρ_{11} ρ_{12} ρ_{13} ρ_{22} ρ_{23} ρ_{33} is added to the **node_dof** records. See also **group_materi_plasti_kinematic_hardening**. Condition: the **-materi** differential **group_type** should be present.

5.16 materi_strainenergy

The material strain energy $0.5\sigma_{ij}\epsilon_{ij}{}^{\mathrm{elas}}$ is added to the **node_dof** records. You can print or plot it to see where energy is stored after loading. Also **materi_stress** and **materi_strain_elasti** should be initialised.

5.17 materi_strain_elasti

The elastic strain $\epsilon_{kl}{}^{\mathrm{elas}}$ is added to the **node_dof** records. Condition: the **-materi** differential **group_type** should be present. See also: **materi_strain_total**.

5.18 materi_strain_intergranular

The intergranular strain S_{ij} is added to the **node_dof** records. This can be used by hypoplasticity laws, see the theory section.

5.19 materi_strain_plasti

The plastic strain $\epsilon_{kl}{}^{\mathrm{plas}}$ is added to the **node_dof** records. Condition: the **-materi** differential **group_type** should be present. See also: **materi_strain_total**.

5.20 materi_strain_total

The total strain ϵ_{kl} is added to the **node_dof** records. All strains are time integrals of the strain rates for a specific material particle which happens to be present in the node. Condition: the **-materi** differential **group_type** should be present.

5.21 materi_stress

The stresses σ_{11} σ_{12} σ_{13} σ_{22} σ_{23} σ_{33} are added to the **node_dof** records. Condition: the **-materi** differential **group_type** should be present.

5.22 materi_velocity

The velocities v_i are added to the **node_dof** records. Condition: the **-materi** differential **group_type** should be present.

5.23 materi_velocity_integrated

The integrated velocities vi_i are added to the **node_dof** records. The integration of nodal velocities in fact results in displacements. But asking for these integrated velocities doesn't activate automatically that the calculation is done over the total deformed volume (as is the case when you initialize **materi_displacement**), and not automatically a total lagrange model is used in stress analysis.

Condition: also **materi_velocity** should be initialized because these are necessary for calculation **materi_velocity_integrated**.

5.24 materi_void_fraction

The material void fraction f^* is added to the **node_dof** records. This is required for the **group_materi_plasti_gurson** model.

5.25 materi_work

The material second order work $\dot{\sigma}_{ij}\dot{\epsilon}_{ij}$ is added to the **node_dof** records. You can print or plot it to see where material instabilities are present.

5.26 residue

A scalar unknown is added to the **node_dof** records. The scalar is the residue in the differential equations. Also **derivatives** should be initialized.

5.27 wave_scalar

Scalar in wave equation is **node_dof** records. Condition: also **wave_fscalar** should be initialized and the **-materi** differential **group_type** should be present.

5.28 wave_fscalar

The first time derivative in the wave equation is added to the **node_dof** records. Condition: also **wave_scalar** should be initialized and the **-materi** differential **group_type** should be present.

5.29 end_initia (last record of initialization part)

6 Input file, data part

Data items

Data items in the data part are used to control the calculation, select required output, give unknowns initial values, etc.. Note that an **end_data** record is needed.

data_item index data_values
...
data_item index data_values
end_data

Consider the following example

element 0 -tria3 0 1 2
element 1 -tria3 1 2 3
node 0 0. 0.
node 1 1. 0.
node 2 0. 1.
node 3 1. 1.
...
end_data

Note that the data items **element** and **node** are indexed. In fact most data items need to be indexed. Indexing starts at 0 (all numbering in TOCHNOG starts at 0). Indices need not strictly be sequential (e.g. only the indices 1,2 and 5 of a data item may be specified).

Ranges

Ranges are general input formats used for indices and data values. Possible ranges are illustrated by the following examples

12
-all
-ra part1 part2 ...**-ra**

The **-all** range is not available for indices. The **-ra** are delimiters of complex ranges. Each complex range part, e.g. part1, consists by example out of

35
-from 5 -to 16
-from 5 -to 25 -step 2

The data values for a data item can be specified as a range if this is allowed for in the description of the data item. All words in the data part (or part of an index) need to be preceeded with a 'tic' (-). In the example the **node_dof** records 1 to 100 are initialized

node_dof -ra -from 1 -to 100 -ra 1. 0. 0.

Define blocks

You can define a word to represent a set of strings. For each word defined, you need to specify a **start_define** ... **end_define** block. Within the block, you first specify the word, and then you specify the set of strings. Later in the data part, you can use the defined words as the replacement of the set of strings.

Example:

start_define
velocity 1.34
end_define
start_define
left_edge geometry_line 1
end_define
...
left_edge 0. 0. 0. 10. 1.e-4
...
bounda_unknown 1 -left_edge -velx
bounda_time 1 0. 0. 100. velocity
...

The words **plus**, **minus**, **multiply** and **divide** as used in arithmetic blocks are prohibited in define blocks.

Arithmetic blocks

You also can use the arithmetic expressions **plus**, **minus**, **multiply** and **divide**. We show some examples:

(make **A** equal to 4.1)
start_arithmetic
A 1.1 plus 3
end_arithmetic
...
(make **B** equal to 3.2)
start_arithmetic
B 3.2
end_arithmetic
...
(make **C** equal to 7.6)
start_arithmetic
C A plus B
end_arithmetic
...
(make **D** equal to 15.2)
start_arithmetic
D A plus B multiply 2.

end_arithmetic

Expressions will be evaluated from left to right. Words from define blocks will *not* be recognized in arithmetic blocks.

Conditional blocks

Parts of the input file can be processed conditionally within **start_if** ... **end_if** blocks. The is illustrated below with an example:

Example:

start_define
do_complete_calculation true
end_define
...
start_if do_complete_calculation
...
end_if
...

The part in the **start_if** ... **end_if** block is only done if **do_complete_calculation** is set to **true**, like in the example. If **do_complete_calculation** is set to **false** that part will be skipped.

Automatic counting: the counters

The words **counter_a**, **counter_b**, **counter_c** and **counter_d** are reserved words in the data part. If they are found, they will be substituted by their integer value. Initially the value is 0 for both, and each time a counter is encountered it will be increased by 1. The example below shows a typical application.

start_define
left_edge geometry_line counter_a
end_define
start_define
right_edge geometry_line counter_a
end_define
...
left_edge 0. 0. 0. 10. 1.e-4
right_edge 2. 0. 2. 10. 1.e-4
...
bounda_unknown 1 -left_edge -velx
bounda_time 1 0.
bounda_unknown 2 -right_edge -velx
bounda_time 2 1.3
...

Notice that we automatically give the geometry lines a unique number in this way; the unique number is not really of interest in the remainder of the input file, so the application of a counter is convenient.

Finally, also the words **counter_a_apply**, **counter_b_apply**, **counter_c_apply** and **counter_d_apply** are available. They will be substituted by the current value of the counters, without that the counters are incremented.

<u>Control indices</u>

All possible data items are defined in the following sections. It only makes sense to specify some of the data items before the calculation; the other data items are only meant to be printed after the calculation. The example below specifies a 1D temperature calculation.

```
echo -no
number_of_space_dimensions 1
condif_temperature
end_initia

node 1 0
node 2 1
node 3 2

element 1 -bar2 1 2
element 2 -bar2 2 3

bounda_unknown 0 1 -temp
bounda_time 0 0.0 0. 1. 1. 100. 1.
bounda_unknown 1 3 -temp
bounda_time 1 0.0 0.0 100.0 0.

group_type 0 -condif
group_condif_density 0 1.0
group_condif_capacity 0 0.1
group_condif_conductivity 0 0.1
group_condif_flow 0 0.

control_timestep 0 0.1 10.0
control_print 0 -time_current -node_dof
control_print_database 1 -yes
control_timestep 2 0.2 10.0
control_print 2 -time_current -node_dof

end_data
```

Note how the indices of control items like **control_timestep** and **control_print** are used to control the sequence of events. First, (*index=0*) time steps of size 0.1 are taken and for each time step results are printed. Then (*index=1*) the database is printed which can serve as a point of restart. Finally (*index=2*) time steps of size 0.2 are taken and for each time step results are printed.

<u>Numbering of values in records</u>

The numering of values in records in illustrated by **node_dof** records. Look at the following piece of input file

...
number_of_space_dimensions 2
materi_velocity
materi_stress
end_initia
...
node_dof 1 0.0 0.0 -1.0 0.0 0.0 -1.0 0.0 -1.0
node_dof 2 0.0 0.0 -1.0 0.0 0.0 -1.0 0.0 -1.0
...
end_data

Here **node_dof** records 1 and 2 are initialized. The initial velocities are 0, and for the initial stresses we use $\sigma\_xx = -1$, $\sigma\_yy = -1$ and $\sigma\_zz = -1$. The total list of unknowns in the **node_dof** record is **-velx**, **-vely**, **-sigxx**, **-sigxy**, **-sigxz**, **-sigyy**, **-sigyz** and **-sigzz**.

We refer to **-velx** as the 0'th value in the **node_dof** record, **-vely** as the 1'th value, etc. So printing the history of the **-sigxx** stress of **node_dof** record 1 is obtained by this:

...
control_timestep 10 ...
control_print_history 10 -node_dof 1 2
...
end_data

where the number 2 refers to the **-sigxx** stress. See also the definition of the **control_print_history** record for this. As an alternative, sometimes you can use names in stead of numbers, by example here:

...
control_timestep 10 ...
control_print_history 10 -node_dof 1 -sigxx
...
end_data

<u>Types of unknowns</u>

Some of the unknowns are <u>principal</u> unknowns: these are **materi_velocity**, **condif_temperature**, **groundflow_pressure**, **wave_fscalar**. These are the unknowns which are solved by the equilibrium equations (conservation laws).

The other unknowns, like **materi_stress** and so, follow from these principal unknowns (strains follow from displacement derivatives, stresses follow from strains by material laws, etc.).

Furthermore, for all the unknowns we have <u>primary</u> values, which are the unknowns themselves, and <u>derived</u> unknowns, which are the space and time derivatives of the primary unknowns.

6.1 **area_element_group** *index geometry_entity_item geometry_entity_index element_group*

At the start of the calculations, this record is used to generate **element_group** records. Each element, all of whose nodes are part of the *geometry_item*, will get an **element_group** record with value *element_group*. Please realise that the geometry entity can be a two-dimensional area, a volume, etc.

This option comes handy whenever a part of the domain gets some specific element data. By example, this would be the case if different areas in the structure have different material properties like stiffness, etc.

Beware: any directly specified **element_group** records will be overwritten.

We show here two ways to get different element data in different regions: Both ways give elements with young 1.2 from x=0 to x=1, and elements with young 3.3 from x=1 to x=2.

First way:

```
..
node 1 0.
node 2 1.
node 3 2.
element 1 -bar2 1 2
element 2 -bar2 2 3
element_group 1 0
element_group 2 1
..
group_type 0 -materi
group_materi_elasti_young 0 1.2
group_type 1 -materi
group_materi_elasti_young 1 3.3
..
control_mesh_refine_globally 10 -h_refinement
..
```

Second way:

```
..
node 1 0.
node 2 1.
node 3 2.
element 1 -bar2 1 2
element 2 -bar2 2 3
..
group_type 0 -materi
group_materi_elasti_young 0 1.2
group_type 1 -materi
group_materi_elasti_young 1 3.3
..
geometry_line 1 0. 1. 1.e-4
geometry_line 2 1. 2. 1.e-4
```

```
area_element_group 1 -geometry_line 1 0
area_element_group 2 -geometry_line 2 1
..
```

See also **area_element_group_method**, **area_element_group_sequence_elementgroup** etc.

6.2 area_element_group_method *index method*

Set *method* to **-all** or **-any**. If *method* is set to **-all**, then the corresponding **area_element_group** is applied to elements for which all nodes are inside the specified geometry. If *method* is set to **-any**, then the corresponding **area_element_group** is applied to elements for which any of the nodes is inside the specified geometry. Default *method* is **-all**.

6.3 area_element_group_sequence *index element_0 element_1 ...*

See **area_element_group_sequence_elementgroup**.

6.4 area_element_group_sequence_element *index name*

See **area_element_group_sequence_elementgroup**.

6.5 area_element_group_sequence_elementgroup *index group_0 group_1 ...*

This options works more or less the same as the **area_element_group** option. Read that description first.

With this option however, you can specify what the element group numbers of an area (geometry) will be in time. This allows for an easy modelling of change of material models inside a geometry (that is, inside the elements of that geometry).

This option works in combination with the **area_element_group_sequence_element** etc. records (with the same index).

With **area_element_group_sequence_geometry** you select the area (geometry) for which the time sequence of group numbers should be used.

With **area_element_group_sequence** you select the elements which the time sequence of group numbers should be used.

You can use both **area_element_group_sequence_geometry** and **area_element_group_sequence** to select a combination of elements in a geometry and directly specified element numbers.

With **area_element_group_sequence_time** and **area_element_group_sequence_elementgroup** you select time points at which groups should become active; by example, group_0 becomes active at time_0 etc.

With **area_element_group_sequence_element** you select the name of the elements for which the sequence of time versus group will be used; if this **area_element_group_sequence_element** is not specified then all elements in the area (geometry) will be used.

Remark 1: If you want the stresses, strains, etc. to be reset to 0. when the element group changes, then use a **control_unknown_reset_geometry** record for that.

Remark 2: It is more convenient and clear to use the **start_define end_define** option to define the geometries.

Example:

area_element_group_sequence_geometry 0 -geometry_brick 1
area_element_group_sequence_element 0 -hex8
area_element_group_sequence_time 0 0. 2. 3.
area_element_group_sequence_elementgroup 0 1 5 4

group_type 1 ...
...
group_type 5 ...
...
group_type 4 ...
...
control_unknown_reset_geometry 10 -geometry_brick 1
...

In the selected geometry element group 1 will be used starting from time 0 for elements **-hex8**. Starting from time 2 element group 5 will be used, etc.

Same example, now with define's however:

start_define
soil_empty_wall geometry_brick 1
end_define
...

area_element_group_sequence_geometry 0 -soil_empty_wall
area_element_group_sequence_element 0 -hex8
area_element_group_sequence_time 0 0. 2. 3.
area_element_group_sequence_elementgroup 0 1 5 4

group_type 1 ...
...
group_type 5 ...
...
group_type 4 ...
...
control_unknown_reset_geometry 10 -soil_empty_wall
...

6.6 area_element_group_sequence_geometry *index geometry_entity_item geometry_entity_index*

See **area_element_group_sequence_elementgroup**.

6.7 area_element_group_sequence_method *index method*

Set *method* to **-all** or **-any**. If *method* is set to **-all**, then the corresponding **area_element_group_sequence_geometry** is applied to elements for which all nodes are inside the specified geometry. If *method* is set to **-any**, then the corresponding **area_element_group_sequence_geometry** is applied to elements for which any of the nodes is inside the specified geometry. Default *method* is **-all**.

6.8 area_element_group_sequence_time *index time_0 time_1 ...*

See **area_element_group_sequence_elementgroup**.

6.9 area_node_dataitem *index geometry_entity_item geometry_entity_index data_item_name*

This record is used to generate *data_item_name* records on all nodes located on the specified geometrical entity. The values for the *data_item_name* should be specified in the **area_node_dataitem_double** record for real precision values, or in the **area_node_dataitem_integer** record for integer values (or words).

This allows you to set records like **node_boundary** or so for nodes located in geometries, without finding out yourself which nodes are located on the geometry.

6.10 area_node_dataitem_double *index value_0 value_1 ...*

See **area_node_dataitem**.

6.11 area_node_dataitem_integer *index value_0 value_1 ...*

See **area_node_dataitem**.

6.12 bounda_force *index node_range unknown_0 unknown_1 ...*

States which ones from the list of dof's in which nodes get prescribed nodal forces. The item *node_range* represents a range of node numbers. In stead of a node range also, by example, **-geometry_line 1** can be used, indicating that the nodes on line 1 get the prescribed nodal forces. The items *unknown_0* etc. can be one of the items listed at **dof_label**. However, neither **-dis** and **-scal** can be used.

For a specific *index*, only one of **bounda_force** and **bounda_unknown** can be specified; thus, either Neumann conditions or Dirichlet conditions can be applied to a particular node, but nor both.

<u>Attention</u>: with this option you get the same nodal force on all the specified nodes. If you want to apply a distributed force on a edge, however, you should use **force_element_edge**. That option gives forces consistent with the displacement field, so not necessarily the same for all nodes. By example the nodes on the side of linear elements on a edge get only half the force.

See also: **bounda_time**, **bounda_sine** and **force_element_edge**.

6.13 **bounda_sine** *index start_time freq_0 amp_0 freq_1 amp_1 ...*

The **bounda_unknown** or **bounda_force** record with the same *index* is imposed with the sum of the sine functions; the first sine function has frequency *freq_0* and amplitude *amp_0*, the second sine function has frequency *freq_1* and amplitude *amp_1*, etc.. The sine functions starts at time 0. More general behavior in time can be imposed by using **bounda_time** records. For a specific *index* only one of **bounda_time** and **bounda_sine** can be specified.

As a typical application the response due to the excitation with a frequency spectrum can be analyzed. Just print the relevant response by **control_print_history** and extract the frequency spectrum of that response signal.

The sine loads will be only imposed after *start_time*.

6.14 **bounda_time** *index time load time load ...*

This record specifies a multi linear time-load diagram for the **bounda_unknown** or **bounda_force** record with the same *index*. Between to time points in the diagram, the load is interpolated linearly (ramp function between the two points).

At all times that an unknown is not prescribed in such way, it is free and determined with the governing differential equations. For a specific *index* only one of **bounda_time**, **bounda_time_file**, **bounda_sine** and **bounda_time_user** can be specified.

As a special option, you can specify only one value in the **bounda_time** record if the load is constant over time (so not time-load sets but directly the constant load value).

As a further special option, you can specify no **bounda_time** and no **bounda_sine** at all; then a 0 value is assumed.

6.15 **bounda_time_file** *index switch*

The same multilinear diagram as in the **bounda_time** record can be specified, now however it is given in the file *index*.**bounda**. This allows for a larger number of *time load* sets than possible with the **bounda_time** record, since the value of that record is limited in length by the maximum length that Tochnog uses to store it.

You need to set *switch* to **-yes**.

By example if *index* is 3 then the sets need to be specified in the file **3.bounda**.

6.16 **bounda_time_user** *index switch*

If *switch* is set to **-yes** a user supplied routine for the time-load diagram will be used.

See also the file **user.cc** in the distribution.

6.17 **bounda_unknown** *index node_range unknown_0 unknown_1 ...*

States which unknowns in which nodes get prescribed values by adjustment of the **node_dof** records. The item *node_range* represents a range of node numbers. In stead of a node range also, by example, **-geometry_line 1** can be used, indicating that the nodes on line 1 get the prescribed boundary values. The items *unknown_0* etc. are one of the primary unknowns listed at **dof_label**.

For a specific *index*, only one of **bounda_force** and **bounda_unknown** can be specified (thus either Neumann conditions or Dirichlet conditions).

Example for discrete node forces in y-direction on the nodes on a line:

bounda_unknown 0 -geometry_line 1 -vely
bounda_time 0 0. 0. 1. 1. 100. 1.

Normally you only should specify boundary conditions on principal unknowns (like velocity, temperature, etc.) and not on strain, stresses, etc.!

Specially for velocity (displacement) unknowns, you can prescibe a rotation around either the x-axis, y-axis or z-axis. In 1D you cannot use this record. In 2D you can only specify a rotation around the z-axis. In 3D you can specify each of the three axis.

Example of an x-axis rotation of node 12 with angular velocity of 0.33 [degrees per unit time]:

bounda_unknown 0 12 -rotation_x_axis
bounda_time 0 0.33

For the rotation 0.33 the rotation vector points in the positive x-axis direction.

See also: **bounda_time**, **bounda_sine**, **force_element_edge** and **force_element_volume**.

6.18 change_dataitem *index data_item_name data_item_index data_item_number operat*

With this record you can specify a data item which should be changed over time. The time table should be given in the **change_dataitem_time** table as time-value sets; at least two sets should be specified.

The *operat* determines how the time-value sets are used. If *operat* is set to **-use**, then the value of the time-value sets is directly used. If *operat* is set to **-add**, then the value of the time-value sets is interpreted as a rate of change, so that the value is multiplied with the time step and then added to the old value.

As a typical example you can use this to prescribe the displacement of a contact geometry over time. Below the y-coordinates of a geometry line which is used in the contact algorithm is changed over time:

contact_geometry 0 -geometry_line 1
...

geometry_line 1 0. 10. 2. 10.
...

```
change_dataitem 0 -geometry_line 1 1 -use
change_dataitem_time 0 0. 10. 100. 0.
change_dataitem 1 -geometry_line 1 3 -use
change_dataitem_time 1 0. 10. 100. 0.
...
```

6.19 **change_dataitem_time** *index time value ...*

See **change_dataitem** and **change_dataitem_time_user**.

6.20 **change_dataitem_discrete** *index switch*

If *switch* is set to **-yes** then the changes applied by the **change_dataitem** and **change_dataitem_time** records (with the same index), will be applied at the discrete time points given in **change_dataitem_time**. Between those time points, no interpolation is used.

More precise, the change of the data item will be applied directly after the time point has passed.

If you don't specify this **change_dataitem_discrete** record then interpolation is used.

6.21 **change_dataitem_time_user** *index switch*

If *switch* is set to **-yes** a user supplied subroutine is used in stead of the **change_dataitem_time** table.

See also the **user.cc** routine included in the distribution.

6.22 **check_combination** *index switch*

Tochnog will check for data records which you specify, if other records needed are also specified. If something is wrong an error is generated. If you set *switch* to **-no** this checking is suppressed.

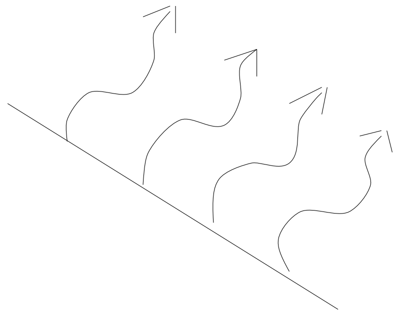
6.23 **condif_convection** *index* α_c T_r

Convection coefficient and convection environmental temperature. Also the record **condif_convection_geometry** should be specified.

6.24 **condif_convection_geometry** *index node_0 node_1 node_2 ...*

Selects the area for which the **condif_convection** record with the same *index* should be applied. This is only available for linear elements.

In 1D, the convective heat flux is 0. In 2D this option is available for all elements. In 3D this option is only available for all linear elements. In stead of a number of nodes also, by example, **-geometry_line 1** can be used in 2D, indicating that the nodes on line 1 start to convect. See also: **condif_convection**.

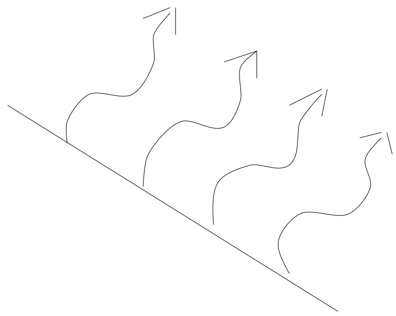


6.25 **condif_radiation** *index* α_r T_r

Radiation coefficient and radiation environmental temperature. Also the record **condif_radiation_geometry** should be specified.

6.26 **condif_radiation_geometry** *index node_0 node_1 node_2 ...*

Selects the area for which the **condif_radiation** record with the same *index* should be applied. This is only available for linear elements.



In 1D, the radiation heat flux is 0. In 2D this option is available for all elements. In 3D this option is only available for linear elements. the triangle between the 3 nodes will radiate heat. In stead of a number of nodes also, by example, **-geometry_line 1** can be used in 2D, indicating that the nodes on line 1 radiate heat. See also: **condif_radiation**.

6.27 **contact_friction** *friction*

See **contact_geometry**.

6.28 **contact_geometry** *index geometry_entity_item geometry_entity_index*

This record specifies a contact geometry. Contacting nodes are forced to stay at the outward normal side of the contact geometry.

The allowed geometries and their material outward normals are listed below

- If a **geometry_point** is used in 1D, the normal is in positive x-direction.
- If a **geometry_line** is used in 2D, the normal is the outer product of 3-direction and the line direction (from point 0 to point 1).

- If a **geometry_circle** is used in 2D, the normal is the outward direction at the circle.
- If a **geometry_ellipse** is used in 2D, the normal is the outward direction at the ellipse.
- If a **geometry_sphere** is used in 3D, the normal is the outward direction at the sphere.
- If a **geometry_polynomial** is used in 2D, the normal is in positive y-direction.
- If a **geometry_polynomial** is used in 3D, the normal is in positive z-direction.
- If a **geometry_triangle** is used in 3D, the normal is in direction of the outer product v01 ∗ v02 where v01 is the vector from node 0 to node 1 and v02 is the vector from node 0 to node 2.
- If a **geometry_quadrilateral** is used in 3D, the normal is in direction of the outer product v01 ∗ v02 where v1 is the vector from node 0 to node 1 and v02 is the vector from node 0 to node 2. Only non-distorted quadrilaterals should be used.

This normal can be switched sign by setting the **contact_geometry_switch** with the same *index* to **-yes**.

In stead of geometries, also contact with all elements will be checked. Only contact with the elements **-bar2**, **-tria3**, **-quad4**, **-tet4** and **-hex8** can be detected. The outer boundary of the elements should be specified by **node_boundary** records. Often it is sufficient to apply **node_boundary** records on the body with the coarsest mesh; if the meshes of the contacting bodies are of the same size however, you should specify **node_boundary** records for both contacting bodies.

The time steps should be such small, that contacter nodes penetrate the other elements only in small steps.

If a **contact_geometry** is used, then the contacter node should also be within the tolerance of the geometrical entity to be noticed!

If contact is detected, normal contact forces of size *contact_penalty_velocity* ∗ penetration are generated between the contacting node and the other element. Moreover, also a frictional force of size *friction* ∗ normal force is generated (see **contact_friction**).

You need more iterations, see **control_timestep_iterations**, then normal; say 5 or more.

As an alternative to the contact algorithm, see **slide_geometry**.

6.29 contact_geometry_switch *index switch*

See **contact_geometry**.

6.30 contact_heatgeneration *factor*

This *factor* specifies how much of the frictional energy is transformed into heat (this only makes sense if *friction* in **contact_friction** is not zero, and if **condif_temperature** is initialized). The *factor* should be between 0 and 1. See also **contact_geometry**.

6.31 contact_penalty_pressure *pressure_penalty*

The *pressure_penalty* should be given some high value if the pressure is freely linked at the surfaces of contacter and target. See also **contact_geometry**.

6.32 contact_penalty_temperature *temperature_penalty*

The *temperature_penalty* should be given some high value if free heat exchange between contacter and target is possible. See also **contact_geometry**.

6.33 contact_penalty_velocity *velocity_penalty*

The **velocity_penalty** essentially puts a spring between the contacter and the target if penetration occurs. Iterations (see **control_timestep_iterations**) are needed; more iterations are needed if the penalty factor is higher. See also **contact_geometry**.

See also **contact_geometry**.

6.34 control_data_delete *index data_item_name index_range*

Delete one or more data items. The *index_range* is a number (e.g. 3) or a range (**-ra ... -ra**, or **-all**). If *index data_item_name* is a nodal item (by example **node** or **node_dof**) then *index_range* can also be a geometrical entity (by example **-geometry_line 1** or so), and the item will be deleted for nodes located on the geometrical entity.

In the example below element 1-10 and nodes 1-100 are deleted after some time in the calculation; this simulates dismantling a part of a structure somewhere in its lifetime. First, time steps with the total structure are taken; then a part of the structure is dismantled; then time steps with the remaining part of the structure are taken.

```
...
control_timestep 10 ...
...
control_data_delete 20 -element -ra -from 1 -to 10 -ra
control_data_delete 21 -node -ra -from 1 -to 100 -ra
...
control_timestep 30 ...
...
```

If an element or node is deleted, then also the corresponding records will be deleted. See also **control_data_put**.

6.35 control_data_put *index data_item_name index_range*

Puts one or more data items. The values to be put should be specified in a **control_data_put_double** record for real data or in a **control_data_put_integer** record otherwise. The *index_range* is a number (e.g. 3) or a range (**-ra ... -ra**, or **-all**).

The **-all** option is only available for nodal data items (like **node** or **node_dof**).

If *data_item_name* is a nodal item then *index_range* can also be a geometrical entity (by example **-geometry_line 1** or so), and the item will be put for nodes located on the geometrical entity.

As a special option for putting nodal items, you can use the **control_data_put_double_node** record in stead of the **control_data_put_double** record. In the **control_data_put_double_node** you can specifying a linear change of the nodal item data over the domain. You accomplish this by specifying in the **control_data_put_double_node** record the nodal item data at ($x = 0$, $y = 0$, $z = 0$), then specify the gradients of the data in x-direction, then specify the gradients of the data in y-direction, and then specify the gradients of the data in z-direction. If 1D you only need to specify everything in x-direction, etc. Typcally this can be used for generation a linear initial stress distribution in the domain (geotechnics), or so.

If the data item already exists it is overwritten; else a new record will be generated.

See also **control_data_delete**.

6.36 control_data_put_double *index ...*

See **control_data_put**.

6.37 control_data_put_double_node *index ...*

See **control_data_put**.

6.38 control_data_put_integer *index ...*

See **control_data_put**.

6.39 control_distribute *index distribution_type data_item_name data_item_number ...*

Add a random number, based on a **-uniform** or **-normal** distribution, to the *data_item_name* records. This is done for the *data_item_number* value in those records (0 for the first value, 1 for the second value, etc.). The *distribution_type* should be set to **-uniform** or **-normal**. Use the **control_distribute_values** record to set the range of the distribution (for **-uniform**) or the variance of the distribution (for **-normal**).

If you specify a group item for *data_item_name*, by example **group_materi_elasti_young** or so, then not the group item record self will be changed, but the item will be changed for the elements which use this record; in this way you can give a random distribution to element data like stiffness, plastic properties, etc.

Several sets of *distribution_type data_item_name data_item_number* can be specified. For each set, you need to specify a corresponding value in **control_distribute_values**.

When a **control_distribute** record is encountered, all previous **control_distribute** records are disregarded.

In the first example, an uniform distribution with range 1.2 is added to the nodal temperatures:

...
materi_velocity
condif_temperature
...
control_distribute 10 -uniform -node_dof -temp

control_distribute_values 10 1.2
...

In the second example, an normal distribution with variance 1.e-3 is added to the y coordinate of the nodes:

...
control_distribute 10 -normal -node 1
control_distribute_values 10 1.e-3
...

In the third example, an normal distribution with variance 1. is added to the young's modulus:

...
control_distribute 10 -normal -group_materi_elasti_young 0
control_distribute_values 10 1.
...

6.40 control_distribute_values *index distribution_value distribution_value*

See **control_distribute**.

6.41 control_eigen *index type_of_eigen_analysis number_of_modes*

With this option you can ask for eigenvalues (and eigenvectors). The eigenvectors are determined for the principal unknowns in the system (like velocity, temperature, etc.). This option is only available after the matrices have been set up with **control_options_solver**. And it is assumed that the matrices used in the eigenmodes determination are symmetric.

With *type_of_eigen_analysis* you can determine which eigen problem will be solved. If *type_of_eigen_analysis* is set to **-matrix**, the eigen modes of the matrix will be solved. Other options for *type_of_eigen_analysis* are presently not available. (To be more clear: presently you cannot solve generalized eigenvalue problems like vibration in mechanics, acoustics, etc.)

The eigenvalues are put in the **control_eigen_values** record. The eigenvectors are put in the **node_eigen** records.

Warning: the eigensolvers we use are not tuned for finite element analysis. So the eigensolvers may be very slow, and large amounts of memory may be consumed.

The eigenvalues and eigenvectors are deleted whenever a new timestep is performed with **control_timesteps**. This is done since in general after a new timestep the matrices will have changed, and so the eigenvalues are not valid anymore.

See also **control_eigen_scale**.

6.42 control_eigen_scale *index largest_component*

With this option you can scale the eigenvectors such that the largest component becomes *largest_component*.

See also **control_eigen**.

6.43 control_eigen_values *index eigen_value_0 eigen_value_1 ...*

See also **control_eigen**.

6.44 control_mesh_adjust_geometry *index geometry_entity_item_0 geometry_entity_index_0 geometry_entity_item_1 geometry_entity_index_1*

The nodes of the geometry entity 0 are replaced such that they neatly follow the boundary of geometry 1. In this way, it is easy to make a mesh with elements precisely in specific regions, if this is required to give separate **element_group** data (e.g. materials) to the geometry and it is too difficult to make the mesh at once ok for this.

Nodes moved in this process automatically get a **node_boundary** record.

The created mesh may be quite distorted. Consider using **control_mesh_remesh**.

6.45 control_mesh_delete_geometry *index geometry_entity_item geometry_entity_index*

All elements which are part of the geometry item are deleted. In this way, it is easy to make a mesh with holes, tunneling systems in ground, etc. Remaining nodes in the geometry, are moved onto the edge of the geometry if the corresponding **control_mesh_delete_geometry_movenodes** record with the same index is set to **-yes**; (otherwise, the remaining nodes are left inside the geometry). For these remaining nodes, automatically a **node_boundary** record will be generated.

For a **geometry_point**, elements inside the tolerance distance of the point will be deleted. For a **geometry_circle**, elements in the total inner area of the circle radius plus its tolerance will be deleted. Likewise for other geometries.

If you combine this record with a **control_timestep** record, then the element will be slowly deleted, starting from a complete element at the start of the timestep up to no element at the end of the timestep. This is accomplished by reducing the nodal forces of the elements slowly to zero; at the end of the timestep, the element is deleted completely. This might be usefull for a better convergence behaviour of the iterative process.

See also **control_mesh_delete_geometry_movenodes**, **control_mesh_delete_geometry_element** and **control_mesh_delete_geometry_elementgroup**.

6.46 control_mesh_delete_geometry_element *index element_name_0 element_name_0 ...*

Only elements with names *element_name_0* etc. will be deleted if the **control_mesh_delete_geometry** (with the same index) is used. By example, *element_name_0* is **-quad4**, **-beam**, etc.

If this record is not specified all elements in the geometry will be deleted.

6.47 **control_mesh_delete_geometry_elementgroup** *index element_group_0 element_group_1 . . .*

Only elements from group *element_group_0* etc. will be deleted if the **control_mesh_delete_geometry** (with the same index) is used.

6.48 **control_mesh_delete_geometry_factor** *index factor_0 factor_1 . . .*

The elements deleted by **control_mesh_delete_geometry** (with the same index), will be deleted by a factor *factor_0* at the start of the timesteps up to a factor *factor_1* at the end of the timesteps. If the **control_mesh_delete_geometry** is not used in cobination with timesteps, then directly *factor_1* will be applied.

If this record is not specified then *factor_0* = 0 and *factor_1* = 1.

6.49 **control_mesh_delete_geometry_movenodes** *index switch*

Determines if remaining nodes inside a deleted geometry, are moved onto the edge of the geometry (**-yes**) or not (**-no**). Moving nodes makes that the element mesh exactly fits the deleted geometry, but may also lead to heavily distorted elements. Default this record is **-yes**.

See also **control_mesh_delete_geometry**.

6.50 **control_mesh_delete_small** *index eps*

At the end of a timestep, an element will be deleted when its volume has become smaller than *eps*.

6.51 **control_mesh_generate_beam** *index element_group geometry_entity_item geometry_entity_index*

The same as **control_mesh_generate_truss**, now for beams however.

6.52 **control_mesh_generate_contactspring** *index element_group geometry_entity_item geometry_entity_index*

Generate **-contactspring** springs for nodes which have the same position in space. This can be used to connect these nodes with spring elements, so to model a contact area. Only nodes located on the specified geometry entity will be used.

The generated springs will get an **element_group** record with value *element_group*. So in that element group you can put the properties of the contact springs.

With the **control_mesh_generate_contactspring_element** record you can set between which elements the contactsprings should be generated. By example use **-quad4** and **-trussbeam** if you want to generate contactsprings between those elements.

6.53 control_mesh_generate_contactspring_element *index element_0 element_1*

See **control_mesh_generate_contactspring**.

6.54 control_mesh_generate_spring1 *index element_group geometry_entity_item geometry_entity_index*

Generate **-spring1** springs for nodes. Only nodes located on the specified geometry entity will be used.

The generated springs will get an **element_group** record with value *element_group*. So in that element group you can put the properties of the springs (see **group_spring_stiffness** etc.).

6.55 control_mesh_generate_spring2 *index element_group geometry_entity_item geometry_entity_index*

Generate **-spring2** springs for nodes which have the same position in space. This can be used to connect these nodes with spring elements. Only nodes located on the specified geometry entity will be used.

The generated springs will get an **element_group** record with value *element_group*. So in that element group you can put the properties of the springs (see **group_spring_stiffness** etc.).

Typically you can use this option to connect meshes which were generated with different **control_mesh_macro** records or so.

6.56 control_mesh_generate_truss *index element_group geometry_entity_item geometry_entity_index*

Generate trusses for nodes which are neigbours in space (that is, for nodes which are connected by an isoparametric finite element). Only nodes located on the specified geometry entity will be used.

The generated trusses will get an **element_group** record with value *element_group*. So in that element group you can put the properties of the trusses (see **group_truss_young** etc.).

Typically you can use this option to put easy trusses somewhere in a mesh with isoparametric elements.

6.57 control_mesh_generate_trussbeam *index element_group geometry_entity_item geometry_entity_index*

The same as **control_mesh_generate_truss**, now for trussbeams however.

6.58 control_mesh_generate_truss_beam_loose *index switch*

This record works together with the **control_mesh_generate_truss**, **control_mesh_generate_beam** and **control_mesh_generate_trussbeam** records.

If *switch* is set to **-yes**, the truss or beam of trussbeam will not be connected to the existing nodes, but new nodes will be generated for the generated element.

Afterwards you can typically connect the truss or beam of trussbeam to the existing mesh with constactsprings, so that the end result is that you can model frictional slip between isoparametric elements and structural elements.

See also **control_mehs_generate_contactspring**.

6.59 control_mesh_generate_truss_beam_macro *index macro_0 macro_1 ...*

This record works together with the **control_mesh_generate_truss**, **control_mesh_generate_beam** and **control_mesh_generate_trussbeam** records.

With *macro_0* etc. you can specify the indices of **control_mesh_macro_*** records. Then the trusses (or beams or trussbeams) will only be generated for nodes coming from the mesh generated by the macro records with the specified indices.

This is handy in case you generate two neighbouring meshes with macro's, and want to generate the elements (trusses or beams or trussbeams) in between these two meshes. Normally, both the meshes would get the extra truss (or ..) in case you use a geometry_line or so to specify that the new elements should be generated between the two meshes (this is so, since the nodes of both meshes are located on the geometry_line). With the present **control_mesh_generate_truss_beam_macro** record however you can specify that the new elements should only be generated by looking at the nodes of some of the meshes, and so no double new elements will be generated in between the two meshes.

6.60 control_mesh_macro *index macro_item element_group n ...*

With this record and the **control_mesh_macro_parameters** record you define a macro region. The macro region will automatically be divided into finite elements.

The type of macro region is defined by *macro_item*. You can set *macro_item* to a **-sphere** (3D), **-cylinder_hollow** (3D), **-brick** (3D), **-rectangle** (2D), **-circle (2D)**, **-circle_hollow** (2D) and **-bar** (1D).

The elements to be generated will get **element_group** *element_group*.

With *n ...* you define how much nodes and elements will be generated. For a **-sphere**, first 8 elements are generated; you need to specify the number of refinements. For a **-cylinder_hollow**, you need to specify the number of nodes in the length direction, the number of nodes over the wall thickness, the number of nodes in circ. direction. For a **-brick**, you need to specify the number of nodes in x-direction, the number of nodes in y-direction and the number of nodes in z-direction. For a **-circle**, you need to specify the number of nodes in a cross-section direction. For a **-circle_hollow**, you need to specify the number of nodes over the wall thickness, the number of nodes in circ. direction. For a **-rectangle**, you need to specify the number of nodes in width direction and the number of nodes in thickness direction. For a **-bar**, you need to specify the number of nodes.

In the following example a sphere is generated, after which the nodes get an initial velocity:

```
...
```

number_of_space_dimension 2
...
end_initia
...
control_mesh_macro 20 -sphere ...
control_mesh_macro_parameters 20 ...
...
control_data_put 30 -node_dof -all
control_data_put_double 30 0. 1. ...
...

6.61 control_mesh_macro_element *index element_type*

With this options you can set the element type which will be generated with **control_mesh_macro** (with the same index). This option is only available in 2d and 3d.

For **element** you can use **-tria3**, **-tria6**, **-quad4** and **-quad9** in 2d. For **element** you can use **-tet4**, **-tet10**, **-hex8** and **-hex27** in 2d.

If this record is not specified then **-bar2** (1d), **-quad4** (2d) or **-hex8** (3d) will be generated.

6.62 control_mesh_macro_parameters *index x y ...*

With this record you can specify the dimensions of the **control_mesh_macro** region.

For a **-sphere**, you need to specify the x, y, z coordinates of the middle of the sphere and the radius of the sphere. For a **-cylinder_hollow**, you need to specify the x, y, z coordinates at the start of the cylinder, the x, y, z coordinates at the end of the cylinder, the middle radius, the wall thickness, the start angle and the end angle in degrees (which allows for an open section). For a **-brick**, you need to specify the x, y, z coordinates at the middle, the length in x-direction, the length in y-direction, and the length in z-direction. For a **-circle**, you need to specify the x, y coordinates of the middle and also the radius. For a **-circle_hollow**, you need to specify the same as for the **circle** and additionally the wall thickness, the start angle and the end angle in degrees (which allows for an open section). For a **-rectangle**, you need to specify the x, y coordinates of the middle, the width and the height respectively. For a **-bar**, you need to specify the x coordinate of the middle and the length respectively.

6.63 control_mesh_macro_set_node_boundary *index switch*

Nodes on the boundary edge of the macro region automatically get a **node_boundary** record if the *switch* is set to **-yes**.

If *switch* is set to **-no** then **node_boundary** records are not set.

Default *switch* is set to **-yes**.

6.64 control_mesh_merge *index switch*

If *switch* is set to **-yes**, then nodes with the same coordinates are merged into one node.

6.65 control_mesh_merge_epscoord *index epsilon*

Distance below which nodes will be merged. Default some small value.

6.66 control_mesh_merge_macro_generate *index macro_0 ...*

This record works together with the **control_mesh_merge** record.

With *macro_0* etc. you can specify the indices of **control_mesh_macro_*** or **control_mesh_generate_*** records. Then the merging will only be done for nodes coming from the mesh generated by the macro or generate records with the specified indices.

6.67 control_mesh_merge_not *index geometry_entity_item geometry_entity_index*

The mesh merging from **control_mesh_merge**, with the same index, will not be use for nodes in the geometry specified by *geometry_entity_item geometry_entity_index*.

6.68 control_mesh_new_mesh *index delta*

If **switch** is set to **-yes** then a new mesh is build.

This is useful whenever the old mesh is very distorted as a result of a large deformation process. The parameter *delta* determines the typical element size, with which the new mesh should be build.

This option tends to make elements at the edge of the domain small.

See also **control_mesh_new_mesh_element**.

6.69 control_mesh_new_mesh_element *index element_type*

With this record you can determine the element type which will be created by **control_mesh_new_mesh**. Allowed types are **-bar2**, **-tria3**, **-quad4**, **-tet4** and **-hex8**. If this record is not specified one of the simplex elements (**-bar2**, **-tria3**, **-tet4**; depending on the number of dimensions) will be generated.

6.70 control_mesh_refine_globally *index refinement_type*

This record activates global mesh refinement or global mesh coarsening. This is not available for **-tria3** and **-tet4** elements. Either *refinement_method* is **-h_refinement** (more of the same elements) or *refinement_method* is **-p_refinement** (higher order elements) or *refine_method* is **-p_coarsen** (lower order elements).

As a special option for the **-h_refinement** method, the format **refine_globally** *index* **-h_refinement** *switch_ξ switch_η switch_ζ* can be used. By example in 1D, only **refine_globally** *index* **-h_refinement** *switch_ξ* should be specified. By example in the **-hex8** element, ξ is the isoparametric coordinate running from the first node to the second node, η runs from the first node the third node and ζ runs from the first node to the fifth node. A isoparametric direction will be refined if the corresponding switch is set to **-yes**. This option allows for refinement in specific directions. It should be used with care however, and only gives proper results if the ξ,η and ζ directions of the elements match.

The **control_mesh_refine_globally** will automatically merge nodes which have the same position in space.

Rules for old and new:

- A new generated element inherits its data items from the old element it is generated from.
- If a new generated node is placed on an old element edge it inherits those data items of the old nodes on that old edge that have a property in common; then arbitrarily the data item of one of the old nodes is taken.
- If a new generated node is placed inside an old element it inherits those data items of the old nodes of that old element that have a property in common; then arbitrarily the data item of one of the old nodes is taken.
- For all new nodes the **node_dof** records are interpolated from the old element nodes **node_dof** records by using the old element interpolation functions.

See also **control_mesh_refine_globally_geometry**.

6.71 control_mesh_refine_globally_geometry *index geometry_entity_item geometry_entity_index*

This record can be used together with the **control_mesh_refine_globally** record with the same index. If all nodes of an element edge is part of the geometrical entity, the new generated nodes are also placed on that geometrical entity. This typically is used to follow curved edges of the domain.

The **control_mesh_refine_locally** will automatically merge nodes which have the same position in space.

6.72 control_mesh_refine_locally *index percentage*

An elements will be refined depending on the size of a solution variable. Typically the solution variable**residue** is used, but others can be chosen via **control_mesh_refine_locally_unknown**.

The percentage of elements which will be refined is specified by *percentage*. Typically *percentage* is 10 or so.

This local mesh refinement is only available for **-bar2**, **-bar3**, **-tria3**, **-tria6**, **-tet4** and **-tet10** elements.

See the rules for old and new at **control_mesh_refine_globally**.

See also **control_mesh_refine_locally_geometry**.

6.73 control_mesh_refine_locally_geometry *index geometry_entity_item geometry_entity_index*

This record can be used together with the **control_mesh_refine_locally** record with the same index. If all nodes of an element edge is part of the geometrical entity, the new generated nodes are also placed on that geometrical entity. This typically is used to follow curved edges of the domain.

6.74 **control_mesh_refine_locally_not** *index geometry_entity_0 geometry_entity_index_0*

The refinement as specified in the **control_mesh_refine_locally** record with the same index, will not be applied on the geometry specified by *geometry_entity_0 geometry_entity_index_0*.

6.75 **control_mesh_refine_locally_only** *index geometry_entity_0 geometry_entity_index_0*

The refinement as specified in the **control_mesh_refine_locally** record with the same index, will only be applied on the geometry specified by *geometry_entity_0 geometry_entity_index_0*.

6.76 **control_mesh_refine_locally_unknown** *index unknown*

With *unknown* you can set which unknown will be used for deciding if an element should be refined. The size of the unknown field will be used.

Possibilities for *unknown* are: **-materi_damage**, **-materi_displacement**, **-materi_plasti_kappa**, **-materi_strain_elasti**, **-materi_strain_plasti**, **-materi_strain_total**, **-materi_stress**, **-materi_velocity**, **-materi_void_fraction** and **-residue**.

As a special option you can set *unknown* to **-nothing**; then an element is refined always.

For finding localization zones (e.g. shear bands) choosing **-materi_strain_plasti** or **-materi_damage** seems to be most robust.

6.77 **control_mesh_remesh** *index switch*

This record causes remeshing if *switch* is set to **-yes**.

Nodes are replaced either such that the mesh becomes more regular (geometric remeshing) or such that the mesh becomes more dense in areas of residues (residue remeshing).

By example, the geometric remeshing can be used after local mesh refinement (**control_mesh_refine_locally**). And by example, the residue remeshing can be used so that nodes follow wave propagation phenomena.

The remeshing can only be applied in combination with time steps, so a **control_timestep** record with the same *index* should be present.

See also **control_mesh_remesh_factor** and **node_boundary**.

6.78 **control_mesh_remesh_factor** *index geometry_factor residue_factor*

With this record you can set the relative usage of geometry remeshing and residue remeshing. As the typical remeshing step (i.e. the distance that a node is relocated) is about 100 times as large in geometry remeshing than in residue remeshing, you should use, say, *geometry_factor* is 0.01 and *residue_factor* 1.0 for both effects to be of about equal size. Some examples are given below.

> (only geometric remeshing)
> **control_mesh_remesh** *index* ...

(only geometric remeshing)
control_mesh_remesh *index* ...
control_mesh_remesh_factor *index* **1. 0.**

(only residue remeshing)
...
residue
end_initia
...
control_mesh_remesh *index* ...
control_mesh_remesh_factor *index* **0. 1.**

(both geometric remeshing and residue remeshing)
...
residue
end_initia
...
control_mesh_remesh *index* ...
control_mesh_remesh_factor *index* **0.05 0.5**

The *residue_factor* can only be larger that 0 if **residue** is specified in the initialization part.

If this **control_mesh_remesh_factor** record is not specified, then *geometry_factor* is 1 and **residue_factor** is 0.

Practical tips:

- Lowering the factors will stabilize the remeshing algorithm (this comes handy when elements tend to collapse, etc.).
- Including some geometrical remeshing (by using an *geometry_factor* unequal 0, say *residue_factor*) stabilizes the remeshing and is almost always needed.

6.79 control_mesh_renumber *index lowest_element lowest_node*

The element numbers are made strictly sequential starting from *lowest_element* and the node numbers are made strictly sequential starting from *lowest_node*. Beware using **control_renumber** in combination with, by example, node numbers in printing of **node_dof** records; use **post_point** records instead.

6.80 control_mesh_split *index switch*

If *switch* is set to **-yes** then each **-quad4** element is split into four **-tria3** elements and each **-hex8** element is split into twelve **-tet4** elements. Further, each **-quad9** element is split into four **-tria6** elements and each **-hex27** element is split into six **-tet10** elements.

See the rules for old and new at **control_mesh_refine_globally**. See also **control_mesh_split_only**.

6.81 control_mesh_split_only *index geometry_entity geometry_entity_index*

If this record is used, the corresponding **control_mesh_split** record will only be applied on the geometry specified by *index geometry_entity geometry_entity_index*.

6.82 control_options_convection *index switch*

If *switch* is set to **-yes**, the convection of a material with respect to the mesh is allowed. If *switch* is set to **-no**, the convection of a material with respect to the mesh is not allowed. This is done for timestep records with the same index.

Default *switch* is set to **-yes**. See also **options_convection**.

6.83 control_options_inertia *index switch*

If *switch* is set to **-yes**, inertia terms are included (material mass, heat capacity, ..). If *switch* is set to **-no**, inertia is not included. This is done for timestep records with the same index.

Default *switch* is set to **-yes**. See also **options_inertia**.

6.84 control_options_relaxation *relax_0 relax_1 . . .*

Relaxation parameters for adjusting unknowns in iterations. This can stabilize the calculation. By example, a relaxation parameter of 0.1 means that the corresponding unknown is now completely updated with the iterative change, but only 10 percent of the change is actually applied in a iteration.

If enough iterations are used, the relaxation parameters with not influence the final solution.

You should specify a relaxation parameter term for each principal unknown which is present in the calculation (see the start of the data part description for a list of principal unknowns; these are velocities, temperature, etc.).

This relaxation done for timestep records with the same index. See also **options_relaxation**.

6.85 control_options_skip_groundflow_materidivergence *index switch*

If *switch* is set to **-yes**, then the material divergence part in the groundflow equation is skipped. This is done for timestep records with the same index.

6.86 control_options_skip_plasticity *index switch*

If *switch* is set to **-yes**, any plasticity data in the input file will be ignored. This is done for timestep records with the same index.

This option is convenient for testing your input file just linear, without the need to outcomment each and every part with plasticity data. See also **options_skip_plasticity**.

6.87 control_options_solver *index solver_type*

If *solver_type* is set to **-diagonal** then only the main diagonal of the system matrix will be used for the solution of all unknowns. This gives the program an explicit like structure. In fact, if **control_timestep_iterations** is set to 1, then a classical explicit finite element program is obtained.

If *solver_type* is set to **-matrix_iterative_bicg** then the complete system matrix will be used for solution of the principal unknowns (see the initialization section for an explanation on principal unknowns). A diagonal Preconditioned BiConjugate Gradient method is applied.

If *solver_type* is set to **-matrix_iterative_petsc** then the complete system matrix will be used for solution of the principal unknowns (see the initialization section). The *PETSc* library with iterative solvers is applied. See also **control_options_solver_petsc_ksptype** and **control_options_solver_petsc_pctype**. The PETSc solver is meant for expert users only. See tochnog/src/makefile how to link the PetSc solvers to Tochnog. The *PETSc* library is developed by Satish Balay, William Gropp, Lois Curfman McInnes and Barry Smith (Mathematics and Computer Science Division, Argonne National Laboratory).

If *solver_type* is set to **-matrix_superlu** then the complete system matrix will be used for solution of the principal unknowns (see the initialization section). The *SuperLu* library with a sequential direct solvers is applied (LU decomposition). The SuperLu solver is meant for expert users only. Also available are **-matrix_superlu_mt** (multithreads based parallel version of *SuperLu*) and **-matrix_superlu_dist** (MPI based distributed parallel version of SuperLu). See tochnog/src/makefile how to link the SuperLu solvers to Tochnog. The *SuperLu* library is developed by James W. Demmel (Computer Science Division, University of California, Berkeley), John R. Gilbert (Xerox Palo Alto Research Center) and Xiaoye S. Li (National Energy Research Scientific Computing Center, NERSC). Beware: if the equations are singular, if you forget boundary conditions or so, superlu tends to give strange problems (segmentation faults or so).

In all cases, the first iteration will give estimates for velocities, displacements, etc. Typically you need at least two iterations to get estimates for strains, stresses, etc.

Even if the equations are linear, using more iterations may change the results. This is because, the strain, stresses, etc. are redistributed due to the continuous field approach. Typically the results become more accurate when using more iterations; but using two or three iterations is ok in most cases.

Even in case of plasticity, damage, or so we use the linear elastic stiffness matrix in both the diagonal solver and the matrix solvers. This gives a stable iteration behavior but requires that enough time steps and/or iterations are used.

If *solver_type* is set to **-none** then only the matrices and right-hand sides are setup, but the equations are not really solved.

For the multithreaded parallel solvers use **options_processors** to set the numbers of processors that you want to use.

Default we use **-matrix_iterative_bicg**.

6.88 control_options_solver_petsc_ksptype *index ksptype*

With this record you can set the Krylov Subspace type for the PETSc solver (so the method in which the iterative method searches for the solution of the equations). Possibilities are: **-richardson**, **-chebychev**, **-cg**, **-bicg**, **-gmres**, **-bcgs**, **-tfqmr**, **-tcqmr**, **-cr**, **-lsqr**, **-preonly**. See **control_options_solver_petsc_pctype** on how to set the preconditioning type.

The PETSc solver needs to be installed separately on your computer. See the Tochnog homepage links where to get PETSc. After installing PETSc, you need to compile Tochnog and link it to the PETSc library; follow the instructions in tochnog/src/makefile.

Running on single processor computers.

For running on a single processor computer run Tochnog like **tochnog beam.dat**.

Running on distributed processor systems (clusters of workstations).

With the PETSc solver you can also run on distributed memory systems. The MPI is used for message parsing. We recommend using the MPICH version of MPI; see the links at the homepage where to get it.

For running on distributed systems, run Tochnog like **mpi -v -np 20 tochnog** if you have the *same* filesystem across all the machines. Otherwise you need to use a procgroup file which tells where in each machine the executable and the data file are. Examples of usage are on page 9 of the mpich guide.ps file. If mpich is used for MPI, then the $MPIHOME/util/machines/machines.$ARCH file contains the hostnames. But if the procgroup file is used, then it takes precedence over machines.$ARCH.

A procgroup file is like:

hostname1 0 /home/xxx/tochnog
hostname2 1 /users/yyy/tochnog
...

Also notice that in calling Tochnog no arguments are specified so always the default input file **tn.dat** is used.

6.89 control_options_solver_petsc_pctype *index pctype*

With this record you can set the Preconditioner type for the PETSc solver. Possibilities are: **-jacobi**, **-bjacobi**, **-sor**, **-eisenstat**, **-icc**, **-ilu**, **-asm**, **-sles** and **-lu**. Especially the **-lu** is noteworthy, since it is a complete LU factorisation so that it activatives a direct solver (sparse, profile).

6.90 control_print *index data_item_name_0 data_item_name_1 ...*

The is the normal printing command. A **control_print** record causes the data items with name *data_item_name_0*, etc. to be printed. Example

control_print 1 -node -node_dof

See also: **print_filter**.

6.91 control_print_database *index switch*

If *switch* is set to **-yes**, the complete database is be printed. See the example below

control_print_database 6 -yes

This database contains the complete status of the calculation. By example if *index* is 6, the data base is printed in the file input_file_name6.dbs.

As a special option, you can print a restart database by setting *switch* to **-restart**. This database is essentially the same as the complete database but, however, all control records up to the present *index* are deleted since these do not need to performed anymore in a restart.

Always a complete database tn.dbs is printed at the end of the calculation.

6.92 control_print_data_versus_data *index data_item_name_0 index_0 number_0 data_item_name_1 index_1 number_1 ...*

This option prints columns of data for each time step. Print in the first column the *number_0* value of *data_item_name_0* with index *index_0*. Similar in the second column for *data_item_name_1 index_1 number_1*. Etc. (for all values).

Typically, the data item names can be **-node_dof** such that unknowns can be printed against each other in time. If the data item names are **-node_dof**, then *number_0* and *number_1*, etc. can be **-velx**, **-temp**, etc. (see **dof_label**); otherwise, by example, if *number_0* is 3 then the fourth value of *data_item_name_0* is printed. All results will be printed in the file tn.dvd.

Example:

control_print_data_versus_data 0 -node_dof 2 -temp
-node_dof 2 -sigxx -node_dof 2 -sigxx

Another example:

post_point 0 0.0 1.0
post_calcul -materi_stress -average -materi_stress -sizedev

control_print_data_versus_data 20
-time_current 0 0
-post_point_dof_calcul 0 0 -post_point_dof_calcul 0 1

In the last example, the **-post_point_dof_calcul 0 0** stands for 'the **post_point_dof_record** with index 0 and the 0'th number which is the first value so the avarage of the stresses'.

See also: **control_print**.

6.93 control_print_dx *index switch*

If *switch* is set to **-yes** then results are written in a format for IBM data explorer. By example, if the input file is called **turbine.dat** then the results are written in the **turbine.dx**.

Presently, in data explorer do the following to visualize:

- start dx
- import data, data explorer file, select the tochnog dx file
- visualize data

Set the option rendering options to hardware for better results. You are urged to read the quick reference manual for IBM dx. Expect Tochnog specific visualisation networks for data explorer to appear once.

6.94 **control_print_element** *index data_item_name*

With this options you can print values from element data versus coordinates. Select either **-element_truss_force** or **-element_beam_moment** for *data_item_name*.

By example, the first value of a **-element_beam_moment** record (that is number 0 in tochnog notation) will be printed in the file **element_beam_moment_0**.*index*; the file will contain lines containing x, y, z and value. In 1D only x will be printed, etc.

6.95 **control_print_filter** *index print_filter_index_0 print_filter_index_1 ...*

See **print_filter**.

6.96 **control_print_gid** *index switch*

Print the mesh and the unknowns in a file which can be plotted with the GID pre-post processor if *switch* is set to **-yes**. By example, if the input file is called turbine.dat then the mesh is written in the turbine.flavia.dat (2D) or the turbine.flavia.msh file (3D). The results are written in the turbine.flavia.res. If line data is present (truss elements, beam elements, tendons), these will be written to lines.bon.

The mesh and results for unknowns will always be written at the end of the calculation.

Nodes and elements will be renumbered when written to the GID files, because GID preferes sequential numbering. You can keep the node and elements numbers in Tochnog the same as the numbers in GID, by applying a **renumber** *index* **1 1** on the mesh you use (thus keeping the numbers in Tochnog sequential starting from 1).

Since GID uses the time as a label for each set of results, you can only print new results to the GID files at unique time points (that is, at the start of the calculation or after the time has increased with the **control_timestep** option).

Since GID gets confused when the number of elements changes between several **control_print_gid** records, Tochnog will only print GID results for the last mesh used.

Coordinates for nodes will be written in the original configuration. If **materi_velocity** is initialized, also a vector **mesh_deform** will be written for GID which contains the deformation between the original mesh configuration and the deformed mesh configuration. Use the **deform mesh** menu in GID, to draw the deformed configuration by applying the vector **deformation** with a factor 1.

If **materi_velocity** is initialized, also a vector **force** will be written for GID which contains the forces corresponding to the velocities (displacements).

Empty elements (that is, elements with a **materi_density** lower than 0.5) will not be plotted,

As a special option, you can set *switch* to **-separate**. Then the mesh and results will be printed in separate files for GID. The option comes handy when the mesh changes during the calculation; GID cannot plot that if the mesh and results are in the same file.

6.97 control_print_gmv *index switch*

Print the mesh and the unknowns in a file which can be plotted with the General Mesh Viewer if *switch* is set to **-yes**. The files are numbered sequentially. By example, the first results are printed in the file gmv001.inp, the second results are printed in the file gmv002.inp, etc. Also results of **node_dof_calcul** records (if present) will be printed.

Always a gmv.inp file will be generated at the end of the calculation.

Only the elements **-tria3**, **-quad4**, **-quad9**, **-tet4**, **-hex8** and **-hex27** are available. The elements **-quad9** and **-hex27** will be mapped to **-quad4** and **-hex8** elements to allow for **gmv** plotting.

TOCHNOG will temporarily renumber elements and nodes when is writes the GMV data. So the numbering in the GMV plots will not correspond with the numbering as it was in the actual calculation.

If present, TOCHNOG element groups will be attributed to GMV material numbers.

If present, TOCHNOG **post_point** records will be attributed to GMV tracers.

As a special option, you can specify the number of an eigen mode instead of **-yes** after **materi_velocity** has been initialized and an eigenvalue analysis has been performed by **control_eigen**. Then the eigenmode for the velocity is added to the nodal coordinates. In this way you can view the mode.

6.98 control_print_history *index data_item_name_0 data_item_index_0 number_0 ...*

Print the time history for each of the sets *data_item_name_0 data_item_index_0 number_0* .... By example, *data_item_name_0* is **-node_dof** so that the development of an unknown can be followed in time. If **-node_dof** is used, *number_0* is one of **-velx**, **-temp**, etc. (see **dof_label**); otherwise, *number_0* should be an integer specifying the number of the value in the record (for instance number 2 means the third value).

The following history is printed in the file temp112.his

control_print_history 0 -node_dof 112 -temp

6.99 control_print_matlab *index switch*

Print x in x.*index*. Print y in y.*index*. Print z in z.*index*. Also the primary unknowns are printed; by example, print **-temp** in temp.*index*. Furthermore, also data generated by **post_calcul** is printed.

Use **matlab** (a trademark of The MathWorks, Inc.) to plot.

By example, if you want to plot temperatures after a 2D calculation with **control_print_matlab 100 -yes**, then do in **matlab** the following:

load x.100
load y.100
load temp.100
fill(x,y,temp)

By example, if you want to plot temperatures after a 3D calculation with **control_print_matlab 100 -yes**, then do in **matlab** the following:

load x.100
load y.100
load z.100
load temp.100
fill3(x,y,z,temp)

6.100 control_print_plotmtv *index switch*

Print the mesh and the unknowns in a file which can be plotted with **plotmtv** if *switch* is set to **-yes**. By example, the first results are printed in the file plotmtv1.dat, the second results are printed in the file plotmtv2.dat, etc. Also results of **node_dof_calcul** records (if present) will be printed.

Only the elements **-bar2**, **-bar3**, **-bar4**, **-bar5**, **-tria3**, **-quad4**, **-quad9**, **-tet4** and **-hex8** are available.

As a special option, you can specify the number of an eigen mode instead of **-yes** after **materi_velocity** has been initialized and an eigenvalue analysis has been performed by **control_eigen**. Then the eigenmode for the velocity is added to the nodal coordinates. In this way you can view the mode.

6.101 control_print_tecplot *index switch*

Print primary unknown field in **node_dof** into the file **tecplot.plt** if *switch* is set to **-yes**. Also results of **node_dof_calcul** records (if present) are printed in the **tecplot.plt** file.

If *switch* is set to **-mesh**, then also the mesh is printed; this will be done automatically however for a new mesh or if the mesh has been changed.

As a special option, you can specify the number of an eigen mode instead of **-yes** after **materi_velocity** has been initialized and an eigenvalue analysis has been performed by **control_eigen**. Then the eigenmode for the velocity is added to the nodal coordinates. In this way you can view the mode.

You can plot this file with the program **tecplot** (a trademark of Amtec Eng., Inc.).

6.102 control_print_unknowns *index switch*

If *switch* is set to **-yes** then results for the primary unknowns will be printed, including also the coordinates at which the results hold.

By example, for the file temp.*index* will contain lines containing x, y, z and **-temp**. In 1D only x will be printed, etc.

Also results for **node_dof_calcul** records will be printed.

6.103 control_print_unknownsrhside *index switch*

If *switch* is set to **-yes** then results for right-hand-side for the the primary unknowns will be printed, including also the coordinates at which the results hold.

By example, for the file temp_rhside.*index* will contain lines containing x, y, z and right-hand-side of **-temp** (that is, heat flux). In 1D only x will be printed, etc.

6.104 control_print_vtk *index switch*

Print the mesh and the unknowns in a file which can be plotted with the Visual Tool Kit, vtk, if *switch* is set to **-yes**. The results are printed in the file tn*index*.vtk. Also results of **node_dof_calcul** records, if present, will be printed.

Only the elements **-bar2**, **-bar3**, **-tria3**, **-quad4**, **-quad9**, **-tet4**, **-hex8** and **-hex27** are available. The elements **-bar3**, **-quad9** and **-hex27** will be mapped to **-bar2**, **-quad4** and **-hex8** elements respectively for **vtk** plotting.

6.105 control_relaxation_condif_temperature *index group_condif_temperature_relaxation*

Relaxation parameters for temperature. See also **control_relaxation_materi_velocity**.

6.106 control_relaxation_groundflow_pressure *index group_groundflow_pressure_relaxation*

Relaxation parameters for groundflow pressure. See also **control_relaxation_materi_velocity**.

6.107 control_relaxation_materi_velocity *index group_materi_velocity_relaxation*

Relaxation parameters for velocity. This parameter only enters the left hand side of the iterative scheme. If enough iterations are used (see **control_timestep_iterations**), the parameters do not influence the final results.

The **control_relaxation ...** options add 'stiffness' to the left hand side of the equations, and so stabilize the iterations. Another possibility is to use **options_relaxation** which reduces iterative changes to the unknowns.

6.108 control_repeat *index number_of_repeats control_index*

If *number_of_repeats* is larger than 0 the calculation repeats from the *control_index*. The value of *number_of_repeats* is decreased with 1.

A first application is to do many time steps, but print only once in a while:

control_timestep 10 1. 100. ...
control_print 20 -node_dof ...
control_repeat 30 80 10

In the latter example, first 100 timesteps are taken, then results for **node_dof** are printed; this is repeated 80 times.

Also, this **control_repeat** can typically be used to perform a number of refinements combined with time stepping to a new, refined, solution. This is done a fixed number of times.

See also **control_repeat_until_item**.

6.109 control_repeat_until_item *index control_index repeat_criterium data_item_name data_item_index data_item_number*

The calculation repeats from the *control_index* until the data item value as specified by *data_item_name data_item_index data_item_number* satisfies *repeat_criterium*.

If *repeat_criterium* is **-change** then repeating is stopped if the value changes less than *tolerance*. The *tolerance* needs to be specified in the **control_repeat_until_tolerance** record (with the same index). It specifies as a percentage the relative change in the data item value above which repeating occurs.

If *repeat_criterium* is **-value** then repeating is stopped if the absolute value is less than *tolerance*. The *tolerance* needs to be specified in the **control_repeat_until_tolerance** record (with the same index).

This can typically be used to perform a number of refinements combined with time stepping to a new solution until some data item doesn't change much anymore.

See also **control_repeat**.

6.110 control_repeat_until_tolerance *index tolerance*

6.111 control_restart *index switch*

If *switch* is set to **-yes** then the calculation continues with the undeformed mesh. The unknowns (in the **node_dof** records) are reset to the initial values. And **time_current** is set to the initial time.

This allows you to calculate some path dependent behavior completely from the start with a refined mesh.

6.112 control_timestep *index step_size time_increment step_size time_increment ...*

These records define sets of time steps of size *step_size* which are to be taken till the time is increased by *time_increment*. In the example below time steps of 0.1 are taken from time 0.0 up to time 1.0. Then time steps of 0.2 are taken up to time 2.0

control_timestep 0 0.1 1. 0.2 1.

6.113 control_timestep_iterations *index number_of_iterations*

This sets a fixed number of equilibrium iterations in each time step (for time steps of the **control_timestep** record with the same index). If *number_of_iterations* is 1 then the time stepping is completely Euler explicit. For larger values, the time stepping becomes implicit. For many iterations, the time stepping is Euler implicit. Typically, *number_of_iterations* is 1,2 or 3. Default *number_of_iterations* is 2.

As an alternative, you can use **control_timestep_iterations_automatic**.

6.114 control_timestep_iterations_automatic *index ratio_criterium maximum_timestep*

After specification of this record, iterations will be performed until *ratio* in **post_node_rhside_ratio** is less then *ratio_criterium*. Typically, set *ratio_criterium* to 0.01 or 0.001.

TOCHNOG will now also adjust the time step size. If few iterations are needed the time step size is increased. If many iterations are needed the time step size is decreased.

The time step specified in **control_timestep** is used as initial step. The time step is not allowed to become higher then *maximum_timestep*. If the time step becomes very low compared to the initial step, time stepping is aborted.

The initial step as specified in **control_timestep**, should be sufficient small so that this automatic algorithm can fulfill the *ratio_criterium* in that initial step.

6.115 control_timestep_iterations_automatic_stop *index switch*

The **control_timestep_iterations_automatic** options normally will stop the calculation if the minimal timestep is reached and the criterium can not be fulfilled.

However, if you set *switch* in **control_timestep_iterations_automatic_stop** to **-yes** then the calculation will continue.

6.116 control_timestep_multiplier *index multiplier*

If this record is specified, each new time step size is *multiplier* * old time step size. The *step_size* as specified in **control_timestep** will only be used as the initial time step.

This option is handy to study physical processes which develop more slowly when time proceeds. A typical example is consolidation analysis in geotechnics.

6.117 control_unknown_freeze *index unknown_0 unknown_1 ...*

The unknowns as specified in this record are freezed to keep their values. By example, *unknown_0* is **temp**, etc.

6.118 control_unknown_reset_geometry *index geometry_item_name geometry_item_index*

Specifies the geometry on which the **control_unknown_reset** record with the same index should be applied. If this geometry is not specified, the **control_unknown_reset** record will be done for all nodes.

6.119 control_unknown_reset_unknown *index unknown_0 unknown_1 ...*

The unknowns as specified in this record are set to a value specified in the **control_unknown_reset_value** record (with the same index). By example, *unknown_0* is **-eptxx**, etc.

As a special option you can use **-all** to reset all unknowns.

As a typical example, you can set strains to zero in a geotechnical calculation, with an **-updated** material description, after the gravity load has been applied. In this way the strains for further deformations can de distinguished more clearly.

6.120 control_unknown_reset_value *index value*

Specifies the value to which unknowns of the **control_unknown_reset_unknown** record are reset. If this **control_unknown_reset_value** record is not specified, then the unknowns are reset to 0.

6.121 dependency_diagram *index unknown_value_0 ... data_item_value_0 ...*

See **dependency_item**.

6.122 dependency_item *index data_item element_group unknown n*

This record allows you to make an element data item dependent on one of the unknowns (see **dof_label** for unknown names). This is done for n values of the unknown (n should be at least 2). The dependency should be specified in the **dependency_diagram** record (same index) with a multi linear diagram. In the diagram first a set of unknowns values should be specified. Second the set of data item values for those unknown values should be specified.

Some examples are given below.

Temperature dependent Young's modulus of element_group 1 ($E = 1.e10$ at temperature 1, etc.):

dependency_item 1 -group_materi_elasti_young 1 -temp 4
dependency_diagram 1 1. 2. 3. 4. 1.e10 1.e9 1.e8 3.e5

Temperature dependent Young's moduli in two maxwell chains of element_group 1 (for the first chain the moduli $1.e10, 1.e9, \ldots$ for the second chain the moduli $1.e10, 1.e9, \ldots$ all relaxation times are $1.\ 10^{-2}$.):

dependency_item 1 -group_materi_maxwell_chain 1 -temp 4
dependency_diagram 1
1. 2. 3. 4.

1.e10 1.e9 1.e8 3.e5
1.e-2 1.e-2 1.e-2 1.e-2
1.e12 1.e11 1.e10 3.e7
1.e-2 1.e-2 1.e-2 1.e-2

As a special option, *unknown* can be set to **-time_current**. This allows for time-dependent properties (aging). The example below shows time dependent Young's modulus of element_group 1 ($E = 1.e10$ at time 0, etc.):

dependency_item 1 -group_materi_elasti_young 1 -time_current 4
dependency_diagram 1 0. 1. 2. 3. 1.e10 1.e9 1.e8 3.e5

The dependencies are available only for real data (and thus not for integer data). The **dependency_diagram** values shows be specified from low to high values for the unknown.

6.123 dof_label *unknown_0 unknown_1 ...*

This record will be filled with labels of the unknowns in the correct order. This information is required for understanding records like **node_dof** etc. The sequential order for the primary unknowns will match the order in which they are specified in the initialization part.

The total list of possible unknown labels is: **-dam** damage, **-dens** density, **-ener** material strain energy, **-epexx** xx-strain elastic, **-epexy**, **-epexz**, **-epeyy**, **-epeyz**, **-epezz**, **-eppxx** xx-strain plastic, **-eppxy**, **-eppxz**, **-eppyy**, **-eppyz**, **-eppzz**, **-eptxx** xx-strain total, **-eptxy**, **-eptxz**, **-eptyy**, **-eptyz**, **-eptzz**, **-eppsize** size of plastic strains, **-gvelx** ground water velocity in x-direction, **-gvely**, **-gvelz**. **-hisv0**, **-hisv1**, ..., material history variables, **-kap** plastic isotropic hardening parameter kappa, **-pres** hydraulic pressure head, **-res** residue unknown, **-rhoxx** xx plastic kinematic hardening, **-rhoxy**, **-rhoxz**, **-rhoyy**, **-rhoyz**, **-rhozz**, **-scal** scalar, **-temp** temperature, **-f** plasticity yield rule, **-fn** nonlocal plasticity yield rule, **-fscal** time derivative of scalar, **-sigxx** xx-stress, **-sigxy**, **-sigxz**, **-sigyy**, **-sigyz**, **-sigzz**, **-1msigxx** xx-stress in first maxwell chain, **-1msigxy**, **-1msigxz**, **-1msigyy**, **-1msigyz**, **-1msigzz**, **-velx** velocity in x-direction, **-vely**, **-velz**, **-velix** integrated velocity in x-direction, **-veliy**, **-veliz**, **-void** material void fraction. **-work** material second order work. Furthermore, **-xvelx** denotes the spatial x-derivative of **-velx** in x-direction, etc.. Finally, **-tvelx** denotes the first time derivative of **-velx**, etc.. The time derivative and the space derivatives are only calculated if **derivatives** is included in the initialization part.

By example, the following might be seen after a print of the database

echo -yes
number_of_space_dimensions 2
derivatives
condif_temperature
end_initia
...
dof_label -temp -xtemp -ytemp -ttemp
...

Or, by example, the following might be seen after a print of the database

echo -yes
number_of_space_dimensions 2
condif_temperature
end_initia
...
dof_label -temp
...

6.124 element *index element_name node_0 node_1 node_2 ...*

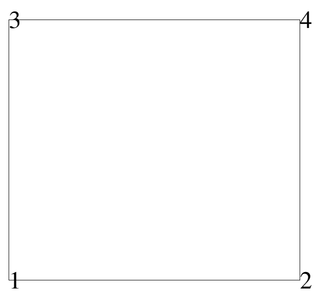
Nodal connective of element *index*. In 1D, *element_name* is **-bar2** (2 noded bar), **-bar3**, **-bar4**.

In 2D, *element_name* is **-tria3** (3 noded triangle), **-tria6** (6 noded triangle), **-quad4** (4 noded quadrilateral), **-quad9**, **-quad16**.

In 3D, *element_name* is **-tet4** (4 noded tetrahedral), **-tet10** (10 noded tetrahedral), **-hex8** (8 noded hexahedral), **-hex27**, **-hex64**.

Example with 4-noded quadrilateral

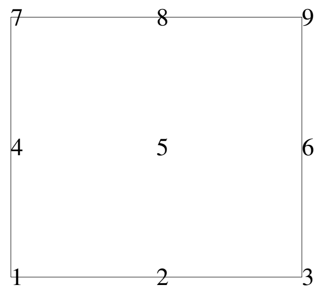
element 0 -quad4 1 2 3 4
node 1 0. 0.
node 2 1. 0.
node 3 0. 1.
node 4 1. 1.



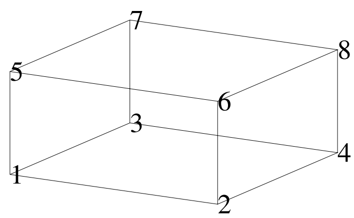
Example with 9-noded quadrilateral

element 0 -quad9 1 2 3 4 5 6 7 8 9
node 1 0.0 0.0
node 2 0.5 0.0
node 3 1.0 0.0
node 4 0.0 0.5
node 5 0.5 0.5
node 6 1.0 0.5
node 7 0.0 1.0
node 8 0.5 1.0
node 9 1.0 1.0

Example with hexahedral



element 0 -hex8 1 2 3 4 5 6 7 8
node 1 0. 0. 0.
node 2 1. 0. 0.
node 3 0. 1. 0.
node 4 1. 1. 0.
node 5 0. 0. 1.
node 6 1. 0. 1.
node 7 0. 1. 1.
node 8 1. 1. 1.



As a special option, user supplied elements can be used by specifying **-user**. See the section on user supplied subroutines.

Except for the **-tria3** and **-tet4** element, the nodal location is always in lobatto points. See also: **group_integration_points**.

Spring elements between two nodes are available by specifying **-spring2** for *element*. The two nodes may have the same position in space for a spring. Spring elements at one node are available by specifying **-spring1** for *element*. The springs are available in 1D, 2D and 3D. You need to specify data like **group_spring_stiffness** etc.

Contact spring elements between two nodes are available by specifying **-contactspring** for *element*. The two nodes may have the same position in space for a spring. The contact springs are available in 1D, 2D and 3D. You need to specify data like **group_contactspring_stiffness** etc.

Truss elements are available by specifying **-truss** for *element*. A truss always has two nodes, which it connects with a elongation stiffness. The trusses are available in 1D, 2D and 3D. You need to specify data like **group_truss_area** etc.

Beam elements are available by specifying **-beam** for *element*. A beam always has two nodes, which it connects with a bending stiffness. The beams are presently only available in 2D. You need to specify data like **group_beam_inertia** etc.

Combined trussbeam elements are available by specifying **-trussbeam** for *element*. This element always has two nodes, which it connects with a elongation stiffness and a bending stiffness. This element is presently only available in 2D. You need to specify data like **group_truss_area**, **group_beam_inertia** etc.

6.125 element_beam_direction *index dir_x dir_y dir_z*

After the calculation, this record will be filled with the direction of a beam in space. The *index* specifies the beam element number.

6.126 element_beam_moment *index force_x_first_node force_y_first_node moment_z_first_node force_x_second_node force_y_second_node moment_z_second_node ...*

After the calculation, this record will be filled with the forces and moments of a beam in the local beam space coordinates x, y, z (such that z is perpendicular to the bending plane of the beam). For a beam in 2d, the local beam x, y, z axes equal the global x, y, z.

The **element_beam_moment** record contains the forces and moments in the first node, and after that the forces and moments in the second node. The *index* specifies the beam element number.

6.127 element_group *index element_group*

This data item is specified which element data items should be taken for the element *index*. Example: elements 0 and 1 get density 1024 while element 2 gets density 1236

element 0 0 1 2
element 1 1 2 3
element 2 2 3 4
...
element_group 0 1
element_group 1 1
element_group 2 2
...
density 1 1024.
density 2 1236.

If no **element_group** records are specified, all element data should use *index* is 0.

See also **area_element_group**.

6.128 element_contactspring_direction *index* $dirN_x$ $dirN_y$ $dirN_z$ $dirT1_x$ $dirT1_y$ $dirT1_z$ $dirT2_x$ $dirT2_y$ $dirT2_z$

After the calculation, this record will be filled with the normal and tangential directions in a contactspring element. The *index* specifies the spring element number.

6.129 element_contactspring_force *index force_N force_T1 force_T1*

After the calculation, this record will be filled with the normal and tangential forces in a contactspring element. The *index* specifies the spring element number.

6.130 element_spring_force *index force*

After the calculation, this record will be filled with the normal force in a spring element. The *index* specifies the spring element number.

6.131 element_tendon_direction *index dir_x dir_y dir_z dir_x dir_y dir_z ...*

This record will be filled with the direction of the tendons which are placed in element *index* (all three directions for each tendon). See also **tendon**.

6.132 element_tendon_number *index tendon_0 tendon_1 ...*

This record will be filled with the tendons which are placed in element *index*. See also **tendon**.

6.133 element_tendon_strain *index eps_0 eps_1 ...*

This record will contain after the calculation the strains of the tendons which are placed in element *index*. See also **tendon**.

6.134 element_tendon_stress *index sig_0 sig_1 ...*

This record will contain after the calculation the stresses of the tendons which are placed in element *index*. See also **tendon**.

6.135 element_tendon_volume *index volume_0 volume_1 ...*

This record will contain after the calculation the volume of the tendons which are placed in element *index*. See also **tendon**.

6.136 element_truss_direction *index dir_x dir_y dir_z*

After the calculation, this record will be filled with the direction of a truss in space. The *index* specifies the truss element number.

6.137 element_truss_force *index force*

After the calculation, this record will be filled with the normal force in a truss element. The *index* specifies the truss element number.

6.138 force_element_edge *index force_0 force_1 ...*

Distributed edge source terms (force terms per unit length in 2D or per unit area in 3D). These distributed terms are translated into equivalent nodal force terms on the edges of elements. This option is meant for 2D and 3D calculations. You should specify a force term for each principal unknown which is present in the calculation (see the start of the data part description for a list of principal unknowns). Also the record **force_element_edge_geometry** should be specified, and optionally the records **force_element_edge_factor** and **force_element_edge_time** can be specified.

6.139 force_element_edge_factor *index* a_0 a_1 $\ldots a_n$

This data item defines a polynomial in space. This polynomial gives a factor which is used as a multiplication factor for **force_element_edge** records (with the same index). In this way, you can obtain coordinate dependent forces.

In 1D the polynomial is $a_0 + a_1 x + a_2 x^2 + \ldots$. In 2D the polynomial is $a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2 + a_6 x^3 + a_7 x^2 y + a_8 xy^2 + a_9 y^3 + \ldots$. In 3D the polynomial is $a_0 + a_1 z + a_2 z^2 + \ldots$.

6.140 force_element_edge_geometry *index node_0 node_1 node_2 ...*

Selects the area for which the **force_element_edge** record with the same *index* should be applied. In stead of specifying node numbers also, by example, **-geometry_line 1** can be used in 2D, indicating that the nodes on line 1 get the distributed force.

Attention: if this **force_element_edge_geometry** option is used INSIDE a FE mesh, then the elements on each side of the geometry will get the force. So you may need to specify only half of the physical force value.

6.141 force_element_edge_sine *index start_time freq_0 amp_0 freq_1 amp_1 ...*

The **force_element_edge** record with the same *index* is imposed with the sum of the sine functions; the first sine function has frequency *freq_0* and amplitude *amp_0*, the second sine function has frequency *freq_1* and amplitude *amp_1*, etc.. The sine functions start at time 0. More general behavior in time can be imposed by using **force_element_edge_time** records. For a specific *index* only one of **force_element_edge_sine** and **force_element_edge_time** can be specified.

The sine loads will be only imposed after *start_time*.

More general time behaviour can be specified with **force_element_edge_time**.

6.142 force_element_edge_time *index time load time load ...*

This record specifies a diagram which contains the factors with which the **force_element_edge** record with the same index is applied. Linear interpolation is used to extend the *time load* values to the intervals between these pairs.

If this record is not specified, and the **force_element_edge_sine** record is not specified, the force is applied at all times with a factor 1.

If no external forces like **force_element_edge_time** are specified, the internal element forces become zero at free edges to satisfy equilibrium. This causes, by example, temperature gradients to become zero at free edges in heat problems.

6.143 force_element_edge_normal *index force*

Distributed normal force in the direction of the outward normal at the edge of a geometry. This distributed term is translated into equivalent nodal force terms on the edges of elements. This option is meant for 2D and 3D calculations. Also the record **force_element_edge_normal_geometry** should be specified, and optionally the record **force_element_edge_normal_time** can be specified.

6.144 force_element_edge_normal_factor *index* a_0 a_1 $\ldots a_n$

This data item defines a polynomial in space. This polynomial gives a factor which is used as a multiplication factor for **force_element_edge_normal** records (with the same index). In this way, you can obtain coordinate dependent forces.

In 1D the polynomial is $a_0 + a_1x + a_2x^2 + \ldots$. In 2D the polynomial is $a_0 + a_1x + a_2y + a_3x^2 + a_4xy + a_5y^2 + a_6x^3 + a_7x^2y + a_8xy^2 + a_9y^3 + \ldots$. In 3D the polynomial is $a_0 + a_1z + a_2z^2 + \ldots$.

6.145 force_element_edge_normal_geometry *index geometry_item_name geometry_item_index*

Selects the area for which the **force_element_edge_normal** record with the same *index* should be applied. By example, **-geometry_line 1** can be used in 2D, indicating that the nodes on line 1 get the distributed force.

Attention: if this **force_element_edge_normal_geometry** option is used INSIDE a FE mesh, then the elements on each side of the geometry will get the force. So the total force will normally become zero since the normals of the elements at the side of the geometry are opposite.

6.146 force_element_edge_normal_sine *index start_time freq_0 amp_0 freq_1 amp_1 ...*

Same as **force_element_edge_sine**, now for normal edge loads however.

6.147 force_element_edge_normal_time *index time load time load ...*

This record specifies a diagram which contains the factors with which the **force_element_edge_normal** record with the same index is applied. Linear interpolation is used to extend the *time load* values to the intervals between these pairs.

If this record is not specified, the force is applied at all times with a factor 1.

6.148 force_element_edge_water *index density_water g dir_x dir_y dir_z*

Distributed water pressure force. This distributed term is translated into equivalent nodal force terms on the edges of elements. The distributed force is automatically calculated as density_water g Δz where g is the gravitational acceleration, and Δz is the distance from the phreatic level. In a 3d calculation, the water pressure force acts in the direction dir_x dir_y dir_z. In a 2d calculation, the water pressure force acts in the direction dir_x dir_y (and dir_z should not be specified).

Also the record **groundflow_phreaticlevel** and **force_element_edge_water_geometry** should be specified.

6.149 force_element_edge_water_geometry *index geometry_item_name geometry_item_index*

Selects the area for which the **force_element_edge_water** record with the same *index* should be applied. By example, **-geometry_line 1** can be used in 2D, indicating that the nodes on line 1 get the distributed water pressure force.

Attention: if this **force_element_edge_water_geometry** option is used INSIDE a FE mesh, then the elements on each side of the geometry will get the force. So you may need to specify only half of the physical force value.

6.150 force_element_edge_water_time *index time load time load ...*

This record specifies a diagram which contains the factors with which the **force_element_edge_water** record with the same index is applied. Linear interpolation is used to extend the *time load* values to the intervals between these pairs.

If this record is not specified, the force is applied at all times with a factor 1.

6.151 force_element_volume *index force_0 force_1 ...*

Distributed volume source terms (like force, heat generation, etc.) in differential equations. Here *force_0* is the source term for the first primary unknown, etc. Use the **control_print** data item to print **dof_label** to get the sequential order of the primary unknowns. Specify source terms for each principal unknown which is present in the calculation (see the start of the data part description for a list of principal unknowns). Consider the example with distributed volume force in x-direction for a 2D material:

force_element_volume 0 1.0.

See also **force_element_volume_factor**, **force_element_volume_geometry**, and **force_element_volume_time**.

6.152 force_element_volume_factor *index* a_0 a_1 $\ldots a_n$

This polynomial gives a factor which is used as a multiplication factor for **force_element_volume** records (with the same index). In this way, you can obtain coordinate dependent forces.

In 1D the polynomial is $a_0 + a_1 x + a_2 x^2 + \ldots$. In 2D the polynomial is $a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2 + a_6 x^3 + a_7 x^2 y + a_8 xy^2 + a_9 y^3 + \ldots$. In 3D the polynomial is $a_0 + a_1 z + a_2 z^2 + \ldots$.

6.153 force_element_volume_geometry *index element_0 element_1 element_2 ...*

Restricts the area for which the **force_element_volume** record with the same *index* should be applied. In stead of a specifying element numbers also, by example, **-geometry_quadrilateral 1** can be used in 2D, indicating that the elements on quadrilateral 1 get the distributed force.

6.154 force_element_volume_sine *index start_time freq_0 amp_0 freq_1 amp_1 ...*

Same as **force_element_volume_sine**, now for volume loads however.

6.155 force_element_volume_time *index time load time load ...*

This record specifies a multi-linear diagram which contains the factors with which the **force_element_volume** record with the same index is applied.

If this record is not specified, the force is applied at all times with a factor 1.

6.156 force_gravity *g_x g_y g_z*

Gravitational acceleration.

In 1D, only the gravity in x-direction needs to be specified. In 2D, the gravity in x-direction and y-direction needs to be specified. In 3D, the gravity in x-direction, y-direction and z-direction needs to be specified.

See also **force_gravity_time**.

6.157 force_gravity_time *time load time load ...*

This record specifies a multi-linear diagram which contains the factors with which the **force_gravity** record is applied. This allows you to impose the gravity on a structure slowly, which might be needed for path dependent problems.

If this record is not specified, the gravity is applied at all times with a factor 1.

6.158 geometry_bounda_factor *index factor_0 ...*

This sets, for either the **geometry_line** or the **geometry_triangle** or the **geometry_quadrilateral** with the same index extra factors which are used for the **bounda_unknown** and the **bounda_force** records. The boundary conditions at a node being part of the geometrical entity are multiplied with the interpolation of the factors. For a **geometry_line** either 2 or 3 factors should be specified; 2 factors define a linear variation where the factors hold at the start and end of the line respectively; 3 factors define a parabolic variation where the factors hold at the start, at the middle and at the end of the line respectively. For a **geometry_triangle** 3 factors should be specified (a linear variation with factors for the first, second and third corner point respectively). For a **geometry_quadrilateral** 4 factors should be specified (a linear variation with factors for the first, second, third and fourth corner point respectively). In the example below, node 2 will get temperature $20 * 1.6$ and node 3 will get temperature $20 * 2.2$.

...
number_of_space_dimensions 2

condif_temperature
. . .
end_initia
node 2 0.2 0
node 3 0.4 0.
. . .
geometry_line 1 0. 0. 1. 0. 0.01
geometry_bounda_factor 1 1. 4.
bounda_unknown 0 -geometry_line 1 -temp
bounda_time 0 0. 20. 1.e6 20.
. . .
end_data

6.159 geometry_bounda_sine_x *index a b*

This sets, for the geomerical entity with the same index an extra factor which is used for the **bounda_unknown** and the **bounda_force** records. The factor gives a sinus variation in x-direction. The size of the factor is $sin(a + b * x)$.

6.160 geometry_bounda_sine_y *index a b*

This sets, for the geomerical entity with the same index an extra factor which is used for the **bounda_unknown** and the **bounda_force** records. The factor gives a sinus variation in y-direction. The size of the factor is $sin(a + b * y)$.

6.161 geometry_bounda_sine_z *index a b*

This sets, for the geomerical entity with the same index an extra factor which is used for the **bounda_unknown** and the **bounda_force** records. The factor gives a sinus variation in z-direction. The size of the factor is $sin(a + b * z)$.

6.162 geometry_brick *index x_c y_c z_c l_x l_y l_z tolerance*

This data item defines a brick in space. Other data items can check if nodes are located on this geometry. The coordinate of the center is *x_c y_c z_c*. The length in respectively x, y and z direction are *l_x l_y l_z*. All **node** within a distance *tolerance* are considered to be part of the brick.

6.163 geometry_circle *index x_c y_c radius tolerance*

This data item defines a circle in space. Other data items can check if nodes are located on this geometry. The coordinate of the center is *x_c y_c*. The circle can only be used in 2D. All **node** within a distance *tolerance* of the radius are considered to be part of the circle.

6.164 geometry_circle_segment *index x_c y_c radius side_x side_y tolerance*

This data item defines a circle segment in space. Other data items can check if nodes are located on this geometry. The coordinate of the center is *x_c y_c*. If *side_x* is set to a positive value, say

+1., then only x-values larger then *x_c* are considered to be part of the geometry. If *side_x* is set to a negative value, say -1., then only x-values smaller then *x_c* are considered to be part of the geometry. If *side_x* is set to 0 , then all x-values are considered to be part of the geometry. Likewise remarks hold for y-values. All **node** within a distance *tolerance* of the radius are considered to be part of the circle segment.

6.165 geometry_cylinder *index x_0 y_0 z_0 x_1 y_1 z_1 radius tolerance*

This data item defines a cylinder segment in space. Other data items can check if nodes are located on this geometry. The coordinate of the center point at the bottom is *x_0 y_0 z_0*. The coordinate of the center point at the top is *x_1 y_1 z_1*. The cylinder can only be used in 3D. All **node** within a distance *tolerance* of the radius are considered to be part of the cylinder.

6.166 geometry_cylinder_segment *index x_0 y_0 z_0 x_1 y_1 z_1 radius side_x side_y side_z tolerance*

This data item defines a cylindrical segment in space. Other data items can check if nodes are located on this geometry. The coordinate of the center point at the bottom is *x_0 y_0 z_0*. The coordinate of the center point at the top is *x_1 y_1 z_1*. If *side_x* is set to a positive value, say +1., then only x-values larger then *x_c* are considered to be part of the geometry. If *side_y* is set to a negative value, say -1., then only x-values smaller then *x_c* are considered to be part of the geometry. If *side_x* is set to 0 , then all x-values are considered to be part of the geometry. Likewise remarks hold for y- and z-values. The cylinder segment can only be used in 3D. All **node** within a distance *tolerance* of the radius are considered to be part of the cylinder.

6.167 geometry_ellipse *index x_c y_c a b tolerance*

The coordinate of the center is *x_c y_c*. The equation for the ellipse is:

$$\left(\frac{x - x\_c}{a}\right)^2 + \left(\frac{y - y\_c}{b}\right)^2 = 1$$

Other data items can check if nodes are located on this geometry. The ellipse can only be used in 2D. All **node** within a distance *tolerance* of the ellipse are considered to be part of the ellipse.

6.168 geometry_line *index x_0 y_0 z_0 x_1 y_1 z_1 radius*

This data item defines a line in space. Other data items can check if nodes are located on this geometry. Coordinates of the end points are denoted by *x_0*, etc.. In 1D, only the x-coordinates should be specified, etc.. All **node** within a distance *radius* are considered to be part of the line.

In the example, a line in 2D space is defined and is used by a **convection_geometry** record (nodes located on the line will convect heat)

...
geometry_line 2 1. 0. 1. 1. 0.01
...
group_condif_convection_geometry 0 -geometry_line 2

...

6.169 geometry_point *index x y z radius*

This data item defines a point in space. Other data items can check if nodes are located on this geometry. The coordinate of the point is *x y z*. In 1D, only *x* should be specified, etc.. All **node** within a distance *radius* are considered to be part of the point.

6.170 geometry_polynomial *index* a_0 a_1 $\ldots a_n$ *x_0 x_1 y_0 y_1 tolerance*

This data item defines a polynomial in space if 2D or 3D. Other data items can check if nodes are located on this geometry. In 2D it is the curve $y = a_0 + a_1x + a_2x^2 + \ldots$. In 3D it is the surface $z = a_0 + a_1x + a_2y + a_3x^2 + a_4xy + a_5y^2 + a_6x^3 + a_7x^2y + a_8xy^2 + a_9y^3 + \ldots$. In 2D $x_0 - x_1$ defines the domain of x. In 3D $x_0 - x_1$ defines the domain of x and $y_0 - y_1$ defines the domain of y. All **node** with a distance (that is the y-distance in 2D or the z-distance in 3D) not more than *tolerance* are considered to be part of the polynomial.

6.171 geometry_quadrilateral *index x_0 y_0 z_0 x_1 y_1 z_1 x_2 y_2 z_2 x_3 y_3 z_3 tolerance*

This data item defines a quadrilateral in space. Other data items can check if nodes are located on this geometry. The coordinates of the corner points are *x_0 y_0 z_0* etc.. In 2D, only *x_0*, *y_0* etc. should be specified etc.. The points of the quadrilateral should be specified in the correct order; the order is clearified in the example below.

All **node** within a distance *tolerance* are considered to be part of the quadrilateral (this gives a brick with thickness $2 * tolerance$). Internally in TOCHNOG, the quadrilateral is divided into two **geometry_triangle**s, which is only approximately true if the quadrilateral is twisted. Example

...
number_of_space_dimensions 2
...
geometry_quadrilateral 0 0. 0. 1. 0. 0. 1. 1. 1.
...

Notice the order in which the points are to be specified.

6.172 geometry_set *index geometry_entity_0 geometry_entity_index_0 geometry_entity_1 geometry_entity_index_1 ...*

This set combines a number of geometrical entities (e.g. **geometry_circle**, **geometry_line**, etc.) into a new entity.

As a special option you can refer to nodes with a **node_boundary** record by specifying **geometry_set** *index* **node_boundary**.

Other data items can check if nodes are located on this geometry.

6.173 **geometry_sphere** *index x_c y_c z_c radius tolerance*

This data item defines a sphere in space. Other data items can check if nodes are located on this geometry. The coordinate of the center is *x_c y_c z_c*. All **node** within a distance *tolerance* of *radius* are considered to be part of the sphere.

6.174 **geometry_sphere_segment** *index x_c y_c z_c radius side_x side_y side_z tolerance*

This data item defines a spherical segment in space. Other data items can check if nodes are located on this geometry. The coordinate of the center is *x_c y_c z_c*. If *side_x* is set to a positive value, say +1., then only x-values larger then *x_c* are considered to be part of the geometry. If *side_x* is set to a negative value, say -1., then only x-values smaller then *x_c* are considered to be part of the geometry. If *side_x* is set to 0 , then all x-values are considered to be part of the geometry. Likewise remarks hold for y- and z-values.

All **node** within a distance *tolerance* of *radius* are considered to be part of the spherical segment.

6.175 **geometry_triangle** *index x_0 y_0 z_0 x_1 y_1 z_1 x_2 y_2 z_2 tolerance*

This data item defines a triangle in space. Other data items can check if nodes are located on this geometry. The coordinates of the corner points are *x_0 y_0 z_0* etc.. In 2D the z coordinates should not be specified. All **node** within a distance *tolerance* are considered to be part of the triangle (this gives a wedge with thickness $2 * tolerance$).

6.176 **groundflow_density** ρ

Density of ground water.

6.177 **groundflow_phreaticlevel ...**

Groundwater level.

In a 1D calculation this record should be given x value of the groundwater level. The groundwater is below that x-value.

In a 2D calculation this record should be given sets of $x - y$ which specify the y level of the groundwater at several x locations; In 2D you need to give the $x - y$ sets as follows:

- specify $x - y$ sets for increasing x

In 3D the phreatic line is specified as follows. Denote the lowest x with x_0, the next higher x with x_1 etc. Denote the lowest y with y_0, the next higher y with y_1 etc. Denote the phreatic level z value for x_i y_j with z_ij. Then give the following:

- x_0 y_0 z_00 x_1 y_0 z_10 etc.
- x_0 y_1 z_01 x_1 y_1 z_11 etc.
- etc.

The sets x, y need to form a regular grid in the x, y plane. In 3d, the number of points in x and y direction resprectively should be set with *nx* and *ny* of the **groundflow_phreaticlevel_n** record.

In nodes above the phreatic level the pressure will be set to zero during the calculation.

As a special option in 2D and 3D, you can specify one value only, which sets a constant phreatic level of that value everywhere. In this special case, you do not need to specify **groundflow_phreaticlevel_n**.

6.178 groundflow_phreaticlevel_n nx ny

See **groundflow_phreaticlevel**.

6.179 groundflow_phreaticlevel_bounda *method*

If *method* is set to **-method1** then the phreatic level gives the following boundary conditions.

The phreatic level is used to automatically give nodes which are located exactly on the phreatic level the boundary condition that the hydraulic pressure head is $\rho g z$. (In fact in 2D y is used in stead of z).

Nodes above the phreatic level will be given a hydraulic pressure head of zero.

Notice that this option only works correct if the groundwater level is located exactly on nodes of the mesh.

If *method* is set to **-method2** then the phreatic phreatic level gives the following boundary conditions.

The phreatic level is used to automatically give nodes which are located exactly on or above the phreatic level the boundary condition that the hydraulic pressure head is $\rho g z\_phreatic$. (In fact in 2D y is used in stead of z).

Nodes above the phreatic level (getting also $\rho g z$) are in fact not important for the groundflow calculation results under the phreatic level.

Notice that this option also works correct if the groundwater phreatic level is not located exactly on nodes of the mesh. However, plots of the hydraulic head are more difficult to understand because nodes above the phreatic level don't get value 0 but get value $\rho g z$ (but, again, this is not important for the real groundwater results).

6.180 group_axisymmetric *index switch*

If *switch* is set to **-yes**, the calculation becomes axi-symmetrical (1D or 2D) for the group *index*. Each specified x coordinate becomes a radius and y becomes the length (=vertical) direction. (The z-direction is the axi-symmetric direction). Specify only non-negative x coordinates, i.e. define the computational domain in the right half-plane.

6.181 group_beam_inertia *index I*

Moment of inertia for for a beam. The *index* specifies the element_group, see **element_group**.

6.182 **group_beam_memory** *index memory_type*

Memory model for beam; either **-updated_without_rotation** or **-updated**. The **-updated** model is a geometrically nonlinear model which takes large beam rotations into account. The *index* specifies the element_group, see **element_group**.

6.183 **group_beam_plane** *index first_direction second_direction*

With this option you can use a 2d beam in a 3d calculation. With *first_direction* and *second_direction* you can set the plane in which the beam will act; set these to either **-x**, **-y** or **-z**; set first the smaller one (so e.g. **-x** before **-y**).

Example of beam acting in x-z plane:

...
number_of_space_dimensions 3
materi_velocity
materi_velocity_integrated
materi_stress
end_initia
...
group_beam_inertia 0 ...
group_beam_plane 0 -x -z
...
end_data

Please realize that the beam mechanically still is a 2d beam (it does not model double bending or torsion); it just can act in a 2d plane in 3d space.

Any element output like **element_beam_direction** and **element_beam_moment** is defined in the plane of the beam.

If this record is not specified for a beam in 3d, then **-x -y** is assumed. In 2d, this record cannot be specified (the beam is always in the x-y plane).

The *index* specifies the element_group, see **element_group**.

6.184 **group_beam_young** *index E*

Young's modulus for a beam. The *index* specifies the element_group, see **element_group**.

6.185 **group_condif_absorption** *index a*

Absorption coefficient. The *index* specifies the element_group, see **element_group**.

6.186 **group_condif_capacity** *index C*

Heat capacity. The *index* specifies the element_group, see **element_group**.

6.187 group_condif_conductivity *index k*

Heat conductivity. The *index* specifies the element_group, see **element_group**.

6.188 group_condif_density *index density*

Density for convection-diffusion equation. The *index* specifies the element_group, see **element_group**.

6.189 group_condif_flow *index* $beta_1$ $beta_2$ $beta_3$

Known flow field. In 1D only $beta_1$ should be specified, etc. The *index* specifies the element_group, see **element_group**.

6.190 group_contactspring_cohesion *index c*

The normal contact force F_N is not allowed to become larger then cohesion c in tension (positive values of F_N). If it would become larger, then the contact is broken, a gap is assumed and the contact force F_N is put to 0. To have really a positive F_N for extension of the contact spring, the order of the two nodes as specified in the **element** record for the contact spring should be correct.

Notice that when you use **control_mesh_generate_contactspring** to obtain the contact spring elements, you are not sure what the first and what the second node of an element will be, and thus you should not use this **group_contactspring_cohesion** record. Otherwise, it is not important what you use as first and second node, so that **control_mesh_generate_contactspring** can be used safely.

If this **group_contactspring_cohesion** is not specified, infinite cohesion is assumed.

The *index* specifies the element_group, see **element_group**.

6.191 group_contactspring_direction *index dir_x dir_y dir_z*

Direction of a contactspring; that is the direction normal to the surface of the body to which the contact spring is connected. If this record is not specified, the direction is taken to be from the first node of the spring to the second node. The direction it used to determine the elongation of the spring normal to the surface (penetration). The *index* specifies the element_group, see **element_group**.

6.192 group_contactspring_memory *index memory_type*

Memory model for contactspring; either **-updated_without_rotation** or **-updated**. The **-updated** model is a geometrically nonlinear model which takes large contactspring rotations into account. The *index* specifies the element_group, see **element_group**.

6.193 group_contactspring_stiffness *index* k_N k_T

Stiffnesses for contact springs. The force F_N in normal direction of the contact spring is determined from $F_N = k_N\ u_N$ where u_N is the normal displacement difference of the two nodes (that is, the displacement of the second node in normal direction minus the displacement of the first node in normal direction). The first tangential force F_{T1} of the contact spring is determined from $F_{T1} = k_T\ u_{T1}$ where u_{T1} is the tangential displacement difference of the two nodes in the first tangential direction; the same is done for the second tangential force. The total tangential force $\sqrt{F_{T1} + F_{T2}}$ cannot exceed $f\ F_N$ with f friction coefficient; then frictional slip occurs and the total tangential force is set to $f\ F_N$ To model continuing stick between two bodies just put the friction coefficient f very high.

In 1D the parameters k_T and f will not be used (but should be specified as dummies nevertheless).

The *index* specifies the element_group, see **element_group**.

See also **group_contactspring_friction** and **group_contactspring_friction_automatic**.

6.194 group_contactspring_friction *index f*

With this record you can specify a fixed friction coefficient for contact springs.

The *index* specifies the element_group, see **element_group**.

See also **group_contactspring_stiffness** and **group_contactspring_friction_automatic**.

6.195 group_contactspring_friction_automatic *index switch*

If *switch* is set to **-yes**, the friction coefficient for contact springs will be determined from the plasticity law angle of neighbouring elements. For a neighbouring **group_materi_plasti_mohrcoul** the friction coefficient f will be set to $f = (2./3.)\tan\phi$ with ϕ the friction angle in the mohr-coulomb law of the neighbouring elements. For a neighbouring **group_materi_plasti_diprisco** the friction coefficient f will be set to a value depending on the parameter γ of that law.

If no neigbouring elements with appropriate material law are found, then f will be set to 0.2.

The *index* specifies the element_group, see **element_group**.

6.196 group_groundflow_capacity *index C*

Capacity for ground water flow. The *index* specifies the element_group, see **element_group**.

6.197 group_groundflow_materidivergence *index switch*

If *switch* is set to **-yes**, then the material divergence part $\frac{\partial v}{\partial x_i}$ in the groundflow storage equation is used in the analysis. If *switch* is set to **-no** it is not used. Default *switch* is **-yes**. The *index* specifies the element_group, see **element_group**.

6.198 group_groundflow_permeability *index pe_x pe_y pe_z*

Permeability coefficient in ground water flow, in each space direction. In 1D you only should specify pe_x, etc. The *index* specifies the element_group, see **element_group**.

6.199 group_integration_method *index method*

Here *method* sets the integration method for bars, quads en hexa elements. You can either set *method* to **-gauss** or **-lobatto**.

If this record is not set, the default method as described in **group_integration_points** is chosen.

6.200 group_integration_points *index type*

Here *type* sets the number of integration points in an element. It should be set to **-normal**, **-minimal** or to **-maximal**.

For **-tria3** elements and **-tet4** elements, the integration point will be located in the middle with **-minimal** integration, or the integration points are located in the nodes with **-maximal** integration.

For **-tria6** elements and **-tet10** elements, the integration point will be located in the in the nodes, irrespective of this **group_integration_points** record.

For other elements, if **-minimal** is used then the number of integration points in a direction is set equal to the number of nodes in the direction minus 1, and gauss integration is used. If for the other elements **-maximal** is used then the number of integration points in a direction is set equal to the number of nodes in the direction, and lobatto integration is used.

Default **-minimal** is used for **-bar2**, **-tria3** and **-tet4** elements; it is default **-maximal** otherwise.

If *type* is set to **-normal**, the default integration will be used.

For third or higher order elements (**-bar4**, **-quad16**, etc.) you always need to use **-minimal** if **materi_velocity** is initialized.

The *index* specifies the element_group, see **element_group**.

6.201 group_materi_damage_mazars *index* $\epsilon_0 a_t$ b_t a_c b_c β

Parameters for the Mazars damage law. The *index* specifies the element_group, see **element_group**.

6.202 group_materi_damping *index d*

Material damping coefficient. The *index* specifies the element_group, see **element_group**.

6.203 group_materi_density *index density*

Density for material flow equation. This may be overruled by specifying the density as an unknown in the initialization part. The *index* specifies the element_group, see **element_group**.

6.204 group_materi_density_groundflow *index density_wet density_dry*

Density for material flow equation when a calculation is performed in combination with groundflow. In case an element is above the phreatic level, as specified by **groundflow_phreaticlevel_bounda**, the *density_dry* will be used. In case an element is below the phreatic level, the *density_wet* will be used.

In case the **groundflow_phreaticlevel_bounda** is not specified, the *density_wet* will be used.

The *index* specifies the element_group, see **element_group**.

6.205 group_materi_elasti_camclay_g *index G*

Elastic data G for the modified CamClay model. The *index* specifies the element_group, see **element_group**.

6.206 group_materi_elasti_camclay_poisson *index* ν

Elastic data ν for the modified CamClay model. This option is alternative to the **group_materi_elasti_camclay_g** option (so, only one of both can be defined). With this option the poisson ratio ν is assumed constant, and is used as follows:

$$G = \frac{3}{2}K(1-2\nu)/(1+\nu)$$

The *index* specifies the element_group, see **element_group**.

6.207 group_materi_elasti_compressibility *index co*

Compressibility for materials. The *index* specifies the element_group, see **element_group**.

6.208 group_materi_elasti_lade *index* B R λ

Elastic data $B-0$, R, λ for the Lade model. The *index* specifies the element_group, see **element_group**.

6.209 group_materi_elasti_poisson *index poisson*

Poisson ratio for solid. The *index* specifies the element_group, see **element_group**.

6.210 group_materi_elasti_transverse_isotropy *index dir_0 dir_1 dir_2* C_{aaaa} C_{bbbb} C_{aabb} C_{abab} C_{bcbc}

Specifies the unique direction (*dir_0 dir_1 dir_2*) and the elastic moduli in the transverse isotropic model. The *index* specifies the element_group, see **element_group**.

6.211 group_materi_elasti_volumetric_poisson *index* ν

See **group_materi_elasti_volumetric_young_values**

6.212 group_materi_elasti_volumetric_young_order *index* n

See **group_materi_elasti_volumetric_young_values**

6.213 group_materi_elasti_volumetric_young_values *index* ϵ_0 σ_0 ϵ_1 σ_1 ...

This is a special record to model the volumetric stress part of a nonlinear material, given the experimental results of a volumetric compression test (compression in one direction, fixed size in other two directions).

The table ϵ_0 σ_0 ϵ_1 σ_1 ... specifies the strain-stress results for the volumetric compression test. Together with the poisson ratio as specified in **group_materi_elasti_volumetric_poisson**, an isotropic law in a nonlinear Young's modulus and a constant poisson ratio is fitted to this experiment. The Young modulues in fact is taken as the polynomial expansion $E_0 + E_1\epsilon + E_2\epsilon^2 + \ldots + E_{n-1}\epsilon^{n-1}$ where n denotes the order of the polynomial expansion (as given in **group_materi_elasti_volumetric_young_order**).

The poisson ratio should be taken very high, say 0.4999999 or so, to ensure that the resulting law only models volumetric stresses. Then afterwards a normal young-poisson isotropic law (**group_materi_elasti_young** and **group_materi_elasti_poisson**) can be added to get an extra deviatoric part.

6.214 group_materi_elasti_young *index* E

Young's modulus for solid material. The *index* specifies the element_group, see **element_group**.

6.215 group_materi_elasti_young_polynomial *index* E_0 $E_1 \ldots$

Polynomial parameters for strain dependent Young's modulus for solid material. See the theory part. The *index* specifies the element_group, see **element_group**.

6.216 group_materi_elasti_young_power *index* $p_0 E_0 \alpha$

Power law Young's modulus for solid material. See the theory part. The *index* specifies the element_group, see **element_group**.

6.217 group_materi_expansion_linear *index* α

Linear expansion coefficient. The *index* specifies the element_group, see **element_group**.

6.218 group_materi_expansion_volume *index* β

Volume expansion coefficient. The *index* specifies the element_group, see **element_group**.

6.219 group_materi_failure_cruching *index threshold delete_time*

If the compression strain in an element exceeds *threshold*, the element is considered to be fail. The element will be slowly deleted. It is totally deleted if the *delete_time* has passed.

The *index* specifies the element_group, see **element_group**.

6.220 group_materi_failure_damage *index threshold delete_time*

If the damage in an element exceeds *threshold*, the element is considered to be fail. The element will be slowly deleted. It is totally deleted if the *delete_time* has passed.

The *index* specifies the element_group, see **element_group**.

6.221 group_materi_failure_plasti_kappa *index threshold delete_time*

If the plastic parameter kappa in an element exceeds *threshold*, the element is considered to be fail. The element will be slowly deleted. It is totally deleted if the *delete_time* has passed.

The *index* specifies the element_group, see **element_group**.

6.222 group_materi_failure_rupture *index threshold delete_time*

If the tensile strain in an element exceeds *threshold*, the element is considered to be fail. The element will be slowly deleted. It is totally deleted if the *delete_time* has passed.

The *index* specifies the element_group, see **element_group**.

6.223 group_materi_failure_voidfraction *index threshold delete_time*

If the void fraction in an element exceeds *threshold*, the element is considered to be fail. The element will be slowly deleted. It is totally deleted if the *delete_time* has passed.

The *index* specifies the element_group, see **element_group**.

6.224 group_materi_hyper_besseling *index* $K_1 K_2 \alpha$

Parameters for Besseling Hyper elastic rubber model. The *index* specifies the element_group, see **element_group**.

6.225 group_materi_hyper_blatz_ko *index* $G \beta$

Parameters for Blatz-Ko model. The *index* specifies the element_group, see **element_group**.

6.226 group_materi_hyper_mooney_rivlin *index* $K_1 K_2$

Parameters for Mooney-rivlin hyper elastic rubber model. The *index* specifies the element_group, see **element_group**.

6.227 group_materi_hyper_neohookean *index* K_1

Parameter for Neo-Hookean hyper elastic rubber model. The *index* specifies the element_group, see **element_group**.

6.228 group_materi_hyper_reducedpolynomial *index* K_1 K_2 ...

Parameters for reduced polynomial hyper elastic rubber model. The *index* specifies the element_group, see **element_group**.

6.229 group_materi_hyper_volumetric_linear *index* K

Parameter for the linear volumetric hyperelasticity model. The *index* specifies the element_group, see **element_group**.

6.230 group_materi_hyper_volumetric_murnaghan *index* $K\beta$

Parameter for the murnaghan volumetric hyperelasticity model. The *index* specifies the element_group, see **element_group**.

6.231 group_materi_hyper_volumetric_ogden *index* $K\beta$

Parameter for the ogden volumetric hyperelasticity model. The *index* specifies the element_group, see **element_group**.

6.232 group_materi_hyper_volumetric_polynomial *index* *K_0* *K_1* ...

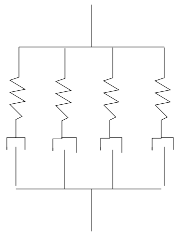
Parameters for the polynomial volumetric hyperelasticity model. The *index* specifies the element_group, see **element_group**.

6.233 group_materi_hyper_volumetric_simotaylor *index* K

Parameter for the simo-taylor volumetric hyperelasticity model. The *index* specifies the element_group, see **element_group**.

6.234 group_materi_maxwell_chain *index E_0 t_0 ... E_n-1 t_n-1*

In total *n* parallel maxwell chains are defined with stiffness *E_0*, relaxation time *t_0*, etc..



The number n should equal **materi_maxwell_stress** in the input initialization part. The *index* specifies the element_group, see **element_group**.

6.235 group_materi_membrane *index switch*

If **switch** is set to **-yes** the zz stress becomes zero in 2D and the yy and zz stress become zero in 1D (in combination with axi-symmetry in 1D, only the yy stress becomes zero since zz is the axi-symmetric direction). If **group_materi_membrane** is not used the plane strain conditions are used. Always the z-thickness is 1. in 3D, and the y, and z-thickness are 1. in 2D; see however also **volume_factor**.

The **group_materi_membrane** option cannot be used in combination with **group_materi_elasti_compressibility**, **group_materi_hyperelasticity** and **group_materi_viscosity**.

The *index* specifies the element_group, see **element_group**.

6.236 group_materi_memory *index memory_type*

Either *memory_type* should be set to **-updated**, **-updated_without_rotation**, **-total**, **-total_piola** or **-total_linear**. See the theoretical part for some explanation.

For a simple linear total Lagrange solid the input file may look like

```
...
materi_velocity
materi_displacement
materi_strain_total
materi_stress
end_initia
...
group_materi_memory 0 -total_linear
group_materi_elasti_young 0 ...
...
end_data
```

For a large deformation total Lagrange solid with a straightforward decomposition of the deformation tensor into a rotation tensor and a stretch tensor the input file may look like

```
...
materi_velocity
materi_displacement
materi_strain_total
```

materi_stress
end_initia
...
group_materi_memory 0 -total
group_materi_elasti_young 0 ...
...
end_data

For a large deformation total Lagrange solid with Piola-Kirchoff stresses and Green-Lagrange strains the input file may look like

...
materi_velocity
materi_displacement
materi_strain_total
materi_stress
end_initia
...
group_materi_memory 0 -total_piola
group_materi_elasti_young 0 ...
...
end_data

For an updated Lagrange solid the input file may look like

...
materi_velocity
materi_strain_total
materi_stress
end_initia
...
node 1 ...
node 2 ...
...
(use -updated_without_rotation to neglect rigid body rotation)
group_materi_memory 0 -updated
group_materi_elasti_young 0 ...
...
end_data

For a fluid the input file may look like

...
materi_velocity
materi_stress
end_initia
...
(use eulerian mesh)
options_mesh -fixed_in_space
...
node 1 ...

node 2 ...
...
group_materi_memory 0 -updated_without_rotation
group_materi_viscosity 0 ...
group_materi_elasti_compressibility 0 ...
...
end_data

The *index* specifies the element_group, see **element_group**.

6.237 group_materi_plasti_boundary *index group_0 group_1*

With this record you can model frictional slip of soil material and alike granular materials on other materials like concrete, steel, etc.

This is done by reducing the friction angle *phi* and diletance angle *phiflow* and cohesion *c* of the granular material with a factor (2./3.). The reduction is done for records **group_materi_plasti_mohrcoul** and **group_materi_plasti_druckprag**, if specified.

The *index* specifies the element_group of the granular material, see **element_group**.

With *group_0* , *group_1* etc. you can specify the groups of the concrete material, steel material etc. The reduction of the friction angle and diletance angle will only be applied to the granular elements (of *element_group*) which are a direct neighbour of an element which has one of the groups *group_0* , *group_1* etc.

Please realise that this method only works well if the finite elements are not too large.

See also **group_materi_plasti_boundary_factor**.

6.238 group_materi_plasti_boundary_factor *index factor*

With this record you can specify a factor other then the default 2./3. used by the **group_materi_plasti_boundary** record.

The *index* specifies the element_group of the granular material, see **element_group**.

6.239 group_materi_plasti_camclay *index* M κ λ

Plastic data M, κ and λ for the modified CamClay model. The *index* specifies the element_group, see **element_group**.

6.240 group_materi_plasti_cap *index* c ϕ α R ϵ_v^p p_b $\ldots$

Plastic data for the modified cap plasticity model. The ϵ_v^p p_b $\ldots$ represents a table with ϵ_v^p versus p_b values; at least two sets of values need to be specified.

The *index* specifies the element_group, see **element_group**.

6.241 group_materi_plasti_compression *index sigy*

Yield data for compression plasticity. The *index* specifies the element_group, see **element_group**. Condition: **materi_strain_plasti** should be initialized.

6.242 group_materi_plasti_diprisco *index* γ $\hat{\beta}_f$ b_p c_p t_p $\hat{\theta}_c$ $\hat{\theta}_e$ ξ_c ξ_e β_f^0 C

Yield data for di Prisco plasticity. The *index* specifies the element_group, see **element_group**. Condition: **materi_strain_plasti** should be initialized.

6.243 group_materi_plasti_druckprag *index phi c phiflow*

Both yield data and flow data (indicated by the word flow) for Drucker-Prager plasticity. Choose *phi* and *phiflow* in between 0 and $\frac{\pi}{2}$. The *index* specifies the element_group, see **element_group**. Condition: **materi_strain_plasti** should be initialized.

6.244 group_materi_plasti_druckprag_tensioncutoff *index switch*

The normal **group_materi_plasti_druckprag** does only limit shear forces, and doesn't care about any tension stresses. That normally isn't a problem since in geotechnical calculations the stress are compressive anyway.

With this **group_materi_plasti_druckprag_tensioncutoff** you can cutoff, however, any tensile stresses beyond the apex. Activate this by setting *switch* to **-yes**.

Activating this cutoff option may lead to a more troublesome convergence behavior.

6.245 group_materi_plasti_gurson *index sigy q1 q2 q3*

Yield data (also used as flow data) for Gurson plasticity. The *index* specifies the element_group, see **element_group**.

6.246 group_materi_plasti_hlc *index sigy m21 m22 m23 m1*

Yield data (also used as flow data) for HLC plasticity. The *index* specifies the element_group, see **element_group**.

6.247 group_materi_plasti_heatgeneration *factor*

This *factor* specifies how much of the plastic energy loss is transformed into heat (this only makes sense if **condif_temperature** is initialized). The *factor* should be between 0 and 1. The *index* specifies the element_group, see **element_group**.

6.248 group_materi_plasti_hypo_cohesion *index c*

Cohesion parameter in hypoplastic law; see the theory section. The *index* specifies the element_group, see **element_group**.

6.249 group_materi_plasti_hypo_intergranularstrain *index* R m_R m_T β_r χ

Intergranular strain parameters in hypoplastic law; see the theory section. The *index* specifies the element_group, see **element_group**.

6.250 group_materi_plasti_hypo_pressuredependentvoidratio *index switch*

If *switch* is set to **-yes** the initial void ratio is corrected for pressure dependency; see the theory section. The *index* specifies the element_group, see **element_group**.

6.251 group_materi_plasti_hypo_wolffersdorff *index* φ h_s n e_{c0} e_{d0} e_{i0} *alpha beta*

Von-Wolffersdorf parameters in hypoplastic law; see the theory section. Here φ is in degrees. The *index* specifies the element_group, see **element_group**.

Attention: you need to initialize 4 history variables for application of this hypoplasticity model. The first history variable contains the void ratio, and should be initialized by initially specifying **node_dof** records. The second history variable will be filled with the time step size of an internally used sub-stepping algorithm for hypoplasticity; this is meant for postprocessing only. The third history variable will be filled with the mobilized friction angle; this is meant for postprocessing only. The fourth history variable will be filled with the a measure of the effective stiffness following from the hypoplasticity law ($\sqrt{M_{ijkl}M_{ijkl}}$). this is meant for postprocessing only.

6.252 group_materi_plasti_kinematic_hardening *index a*

This record specifies the size of the rate of the kinematic hardening matrix ρ_{ij}. The *index* specifies the element_group, see **element_group**.

6.253 group_materi_plasti_matsuokanakai *index phi c phiflow*

Both yield data and flow data (indicated by the word flow) for Matsuoka-Nakai plasticity. Choose *phi* and *phiflow* in between 0 and $\frac{\pi}{2}$. The *index* specifies the element_group, see **element_group**.

6.254 group_materi_plasti_mohrcoul *index phi c phiflow*

Both yield data and flow data (indicated by the word flow) for Mohr-Coulomb plasticity. Choose *phi* and *phiflow* in between 0 and $\frac{\pi}{2}$. The *index* specifies the element_group, see **element_group**.

6.255 group_materi_plasti_mohrcoul_softening *index phi_0 c_0 phiflow_0 phi_1 c_1 phiflow_1 kappa_crit*

Both yield data and flow data (indicated by the word flow) for Mohr-Coulomb softening plasticity. See the theoretical part. Choose each of the angles *phi_0 phiflow_0 phi_1 phiflow_1* in between 0 and $\frac{\pi}{2}$. The *index* specifies the element_group, see **element_group**.

6.256 group_materi_plasti_mohrcoul_tensioncutoff *index switch*

The normal **group_materi_plasti_mohrcoul** does only limit shear forces, and doesn't care about any tension stresses. That normally isn't a problem since in geotechnical calculations the stress are compressive anyway.

With this **group_materi_plasti_mohrcoul_tensioncutoff** you can cutoff, however, any tensile stresses beyond the apex. Activate this by setting *switch* to **-yes**.

Activating this cutoff option may lead to a more troublesome convergence behavior.

6.257 group_materi_plasti_tension *index sigy*

Yield data for tension plasticity. The *index* specifies the element_group, see **element_group**. Condition: **materi_strain_plasti** should be initialized.

6.258 group_materi_plasti_user *index switch*

If *switch* is set to **-yes** the user supplied routine for plasticity is called.

See also the file **user.cc** in the distribution.

The *index* specifies the element_group, see **element_group**.

6.259 group_materi_plasti_visco_always *index switch*

This record specifies if the visco-plasticity yielding, for the viscoplastic law with the same index, is used for all values of the yield function (both positive and negative) or only for positive values of the yield function.

If *switch* is set to **-yes** then viscoplastic yielding is used for all values of the yield function f. If *switch* is set to **-no** then viscoplastic yielding is nonly used for positive values of the yield function f.

If this **group_materi_plasti_visco_always** record is not specified, the *switch* is assumed to be **-no**.

The *index* specifies the element_group, see **element_group**.

6.260 group_materi_plasti_visco_exponential *index* γ α

This record specifies visco-plasticity data for the exponential model. It should be used in combination with a plasticity model.

The *index* specifies the element_group, see **element_group**.

6.261 group_materi_plasti_visco_power *index* η p f^{ref}

This record specifies visco-plasticity data for the power model. It should be used in combination with a plasticity model.

The *index* specifies the element_group, see **element_group**.

6.262 group_materi_plasti_vonmises *index* σ_{y0}

Yield data for Von-Mises plasticity.

The *index* specifies the element_group, see **element_group**. Condition: **materi_strain_plasti** should be initialized.

6.263 group_materi_plasti_vonmises_nadai *index* C κ_0 n

Data for Von-Mises Nadai hardening. The σ_{y0} of the **group_materi_plasti_vonmises** record is taken as σ_{y0} in the nadai law.

The *index* specifies the element_group, see **element_group**. Condition: **materi_plasti_kappa** should be initialized.

6.264 group_materi_stokes *index switch*

If *switch* is set to **-yes**, then stokes flow is used. The *index* specifies the element_group, see **element_group**.

6.265 group_materi_viscosity *index* ν

Viscosity for Newtonian flow. The *index* specifies the element_group, see **element_group**.

6.266 group_materi_viscosity_user *index switch*

If *switch* is set to **-yes**, the user supplied routine for the viscosity for Newtonian flow is used. The *index* specifies the element_group, see **element_group**.

6.267 group_materi_viscosity_bingham *index* σ_y γ m

Parameters for the non-Newtonian bingham viscosity law. The *index* specifies the element_group, see **element_group**.

6.268 **group_materi_viscosity_exponential** *index* ν_0 *m*

Parameters for the non-Newtonian exponential viscosity law. The *index* specifies the element_group, see **element_group**.

6.269 **group_materi_viscosity_heatgeneration** *switch*

If *switch* is set to **-yes**, then viscous dissipation will be used as a heat generation source. See also the theoretical part at the start of this manual. The *index* specifies the element_group, see **element_group**.

6.270 **group_spring_direction** *index dir_x dir_y dir_z*

Direction of a spring. If this direction is specified, it used to determine the elongation of the spring in this direction. If this record is not specified, the direction is taken to be from the first node of the spring to the second node. The *index* specifies the element_group, see **element_group**.

6.271 **group_spring_plasti** *index* F_y

Maximum force in a spring. The *index* specifies the element_group, see **element_group**.

6.272 **group_spring_stiffness** *index* k

Stiffness of a spring. It is multiplied with the elongation of the spring to calculate the spring force. The *index* specifies the element_group, see **element_group**.

6.273 **group_time** *index birth death*

With this option you can set the time of birth of the elements (in group *index*) and the time of death of the elements.

Out of the range *birth* - *death* the elements of the group will not be used in the calculation (the starting *birth* limit itself is not included in the range, whereas the ending *death* limit itself is included).

6.274 **group_truss_area** *index* A

Cross-sectional area for a truss. The *index* specifies the element_group, see **element_group**.

6.275 **group_truss_density** *index* ρ

Density for a truss. The *index* specifies the element_group, see **element_group**.

6.276 **group_truss_memory** *index memory_type*

Memory model for truss; either **-updated_without_rotation** or **-updated**. The **-updated** model is a geometrically nonlinear model which takes large truss rotations into account. The *index* specifies the element_group, see **element_group**.

6.277 **group_truss_rope** *index switch*

The truss will act as a rope if *switch* is set to **-yes**. This means that negative forces will not be allowed (the force remains zero in compression). The *index* specifies the element_group, see **element_group**.

6.278 **group_truss_plasti** *index* σ_y

Yield stress for truss. The actual stress cannot exceed the yield force. The *index* specifies the element_group, see **element_group**.

6.279 **group_truss_young** *index E*

Young's modulus for a truss. The truss force F is $F = EA\Delta u$, where Δu is the elongation of the truss. The *index* specifies the element_group, see **element_group**.

See also **group_truss_young** and **group_truss_area**.

6.280 **group_type** *index type_name_0 type_name_1 ...*

With this record a differential equation is specified for the element group *index*. Allowed type names are **-condif**, **-groundflow**, **-materi**, **-wave**, **-spring**, **-truss**, **-beam** and **-trussbeam**. Also **-empty** is allowed; it indicates that the element is empty.

See also **element_group**.

6.281 **group_user_umat** *index switch*

Set *switch* to **-yes** if you want to use a umat.c routine. The *index* specifies the element_group, see **element_group**.

See also the user supplied subroutines section and **group_user_data**.

6.282 **group_user_data** *index* **...**

Data values which will become available for user supplied subroutines. The *index* specifies the element_group, see **element_group**.

See also: the user supplied subroutines section and **group_user_umat**.

6.283 group_volume_factor *index factor*

In 1D or 2D you can specify the cross-section and thickness respectively, for elements of the element group *index* (see **element_group**).

See also **volume_factor**.

6.284 group_wave_speed_of_sound *index c*

Speed of sound in wave equation. The *index* specifies the element_group, see **element_group**.

6.285 inverse_iterations *number_of_iterations*

Number of iterations for inverse modeling of parameters. In each iteration the total time history as defined by the **control_timestep** records will be applied. Moreover, it will be applied for the current estimate of the parameters and once for each parameter varied a bit (see **inverse_variation**); in this way, the sensitivity of the target data with respect to the parameters is determined. At the end of the iteration, these sensitivities are used to determine new estimates for the parameters. In case of problems consider using **inverse_parameter_step**. See also: **inverse_target**.

6.286 inverse_iteration_number *index number*

This record will be filled with the number of inverse iteration. This comes handy in printing (see **control_print**).

6.287 inverse_parameter *index data_item_name data_item_index data_item_number*

Selects a parameter which is to be determined by inverse modeling. Condition: the parameter value should not be 0. See also: **inverse_target**.

6.288 inverse_parameter_limits *index lower_value higher_value*

Specifies the lower and higher allowed value for the **inverse_parameter** (with the same index). Default no limits are used.

6.289 inverse_parameter_step *index size*

The inverse parameter may be adjusted too much in each inverse iteration, such that the inverse process becomes unstable. This record allows you to set a maximum to the adjustment of the **inverse_parameter** (with the same index). Specifying small values (near 0) for *size* gives a stable but slow inverse process. Default, *size* is 1.0e10.

6.290 inverse_parameter_variation *index variation*

Specifies the factor by which the **inverse_parameter** (same index) is varied a bit to determine the sensitivity of the targets with respect to the parameter. A relative variation is used: (varied factor

= (1+*variation*)*factor. On the one hand, it is of importance to use a variation small enough to get a good approximation of the sensitivity of the targets with respect to the parameter. On the other hand the variation should not be such small that we run into problems with the computer accuracy. Default, *variation* is $1.\ 10^{-4}$. See also **inverse_target**.

6.291 inverse_target *index data_item_name data_item_index data_item_number*

This record, in combination with the **inverse_target_data** record with the same index, allows you to specify target data (e.g. experimental data to determine model parameters or required data for optimization) to be used in the inverse modeling. Parameters will be determined by least squares minimization, i.e. the sum

$$\Sigma(penalty\ (actual\_value - target\_value))^2$$

is minimized (*penalty* and *target_value* are the penalty factor and target value, see **inverse_target_data**). In the example below, the temperature (2.33) at node 3 and the temperature (80.2) at post point 10 were measured and are used for inverse modeling of the conductivity. Both experimental results are imposed with a penalty factor 100 in the inverse modeling. In total 20 iterations are used to estimate the conductivity. The initial estimate for the conductivity needs to be specified via the **group_condif_conductivity** record; while iterating this record will be updated with new estimates.

```
...
condif_temperature
end_initia
...
inverse_target 1 -node_dof 3 -temp
inverse_target_data 1 2.33 100.
inverse_target 2 -post_point_dof 10 -temp
inverse_target_data 2 80.2 100.
inverse_parameter 0 -group_condif_conductivity 0 0
inverse_iterations 0 20
...
end_data
```

6.292 inverse_target_data *index target_value penalty*

Specifies the *target_value* (required value) and *penalty* (penalty factor in least squares minimization) for the **inverse_target** record with the same index. A special format *index lower_value higher_value penalty* is available which allows for putting limits on targets; if the actual value is in between the *lower_value* and *higher_value* it considered to be ok; if the actual value is lower than *lower_value* than the lower value becomes the target value and is imposed with the *penalty* factor; likewise for *higher_value*.

6.293 inverse_target_timestep *index timestep_index*

With this record you can specify at the end of which **control_timestep** record the **inverse_target** record (with the same index) should be evaluated. This is useful if transient information is needed for the inverse modeling. Default, the **inverse_target** is evaluated after the last **control_timestep** record.

6.294 node *index coord_0 coord_1 coord_2*

Coordinates of node *index*. In 1D, only *coord_0* should be specified, etc..

You are not allowed to put free nodes (not attached to any element) in your model. These free nodes will be removed automatically.

6.295 node_boundary *index switch*

The *index* corresponds to a node. The node is assumed to be located on the boundary of the domain if this record is specified. You can set *switch* is set to **-yes**.

For the options **control_mesh_remesh**, **control_new_mesh** and the contact algorithm it is necessary to give all the nodes on the boundary.

These **node_boundary** records can also be referred to by **geometry_set** records.

It is not allowed to give all nodes of one element **node_boundary** records.

6.296 node_damping *index damping_x damping_y damping_z*

This record adds a discrete damper to node *index* in x, y and z direction respectively. In 1D only *damping_x* needs to be specified, etc. The damper will lead to a nodal force of the size $damping\_x * v\_x$ where v_x is the velocity in x direction. The same holds for the y and z direction.

6.297 node_dof *index dof_0 dof_1 ...*

dof_0 dof_1 ... are the degrees of freedom (unknowns) at the node with number *index*. The total number and type of the unknowns depends on the initialization part. Each node has the same unknowns.

Unknowns like pressure, temperature, etc. are primary unknowns. The other unknowns, space derivatives and the time derivative, are not primary unknowns. In the example below, **-temp** is 1., **-xtemp** is 0.2 and **-ttemp** is 0.1 in node 6

```
...
number_of_space_dimensions 1
derivatives
condif_temperature
end_initia
...
node_dof 6 1.0 0.2 0.1
...
```

Default all values in the **node_dof** records are set zero at the start of the calculation. See also: **dof_label** and **post_point**.

6.298 **node_dof_calcul** *index ...*

See **post_calcul**.

6.299 **node_dof_start_refined** *index dof_0 dof_1 ...*

This record will be filled with *dof_0 dof_1 ...*, which are the degrees of freedom (unknowns) as specified at the start of the calculation. at the node with number *index*.

If the mesh has been refined, these start values hold for the refined mesh.

See also **node_dof** and **node_start_refined**.

6.300 **node_eigen** *index unknown_0 unknown_1 ... unknown_0 unknown_1 ...*

Eigenvector for node *index* for the principal unknowns. First the results for the unknowns for the first eigen mode are given. Then the results for the unknowns for the second eigen mode are given. Etc.

Only the principal unknowns of these unknowns have been actually solved in the eigen problem.

See also **control_eigen** and the principal unknowns listed in the initialization part.

6.301 **node_mass** *index mass_x mass_y mass_z*

This record adds a discrete mass to node *index* in x, y and z direction. In 1D only the x-mass needs to be specified, etc. The mass will lead to a nodal force of the size $mass\_x * \dot{v}$ where $\dot{v}\_x$ is the acceleration, and to a gravity force if **force_gravity** is specified. The same holds for the y and z direction.

6.302 **node_rhside** *index rhside_0 rhside_1 ...*

This record will contain after the calculations

$$F_{\text{external}} - (F_{\text{static}} + F_{\text{inertia}} + F_{\text{damping}})$$

Here F_{external} are the external forces resulting from **bounda_force** records, F_{static} are the internal static forces (elastic forces, ..), F_{inertia} are the internal inertia forces (mass, capacity, ..) and F_{damping} are the internal damping forces.

For the temperature equation, this will give the heat flow normal to the outer surface (the heat flux to the environment) at prescribed temperatures. For velocity unknowns, this will give the force vector at prescribed displacements. For the pressure in the ground flow equation, this will give the ground flow to the environment at prescribed pressures.

6.303 **node_start_refined** *index coord_0 coord_1 coord_2*

After the calculation, this record will contain coordinates of node *index* as specified at the start of the calculation. If the mesh has been refined this record with contain the start coordinates for the refined mesh. In 1D, only *coord_0* is filled, etc..

6.304 node_stiffness *index stiffness_x stiffness_y stiffness_z*

This record adds a discrete stiffness to node *index* in x, y and z direction respectively. In 1D only *stiffness_x* needs to be specified, etc. The stiffness will lead to a nodal force of the size $stiffness\_x * u\_x$ where u_x is the displacement in x direction. The same holds for the y and z direction. Condition: also **materi_velocity_integrated** or **materi_displacement** should be initialized.

6.305 options_convection *switch*

If *switch* is set to **-yes**, the convection of a material with respect to the mesh is allowed. If *switch* is set to **-no**, the convection of a material with respect to the mesh is not allowed. This is done for all timesteps.

Default *switch* is set to **-yes**. See also **control_options_convection**.

6.306 options_elementloop *switch*

If *switch* is set to **-no** the evaluation of elements in the element loop will not be done. This is meant for testing purposes only.

6.307 options_inertia *switch*

If *switch* is set to **-yes**, inertia terms are included (material mass, heat capacity, ..). If *switch* is set to **-no**, inertia is not included. This is done for all timestep records.

Default *switch* is set to **-yes**. See also **control_options_inertia**.

6.308 options_matrix_group *switch*

If *switch* is set to **-yes** then the element matrices are stored per group (and not per element). This saves memory. It is only possible to use this, however, if the element matrices in a group are really the same (typically in a linear calculation in a regular mesh with elements of the same size and the same shape per group).

6.309 options_matrix_length *matrix_length*

Tochnog normally allocates much (and enough) memory to store element matrices. Especially in 3D calculations with high order elements, the allocated memory size may be eccesive large. Then you can set *matrix_length* with this **options_matrix_length** record to a low value, which will be used for memory allocation. If this value *matrix_length* would be too small to store the element matrices, you will be automatically warned and asked to make the value higher.

6.310 options_mesh *specifier_x specifier_y specifier_z*

If *specifier_x* is set to **-fixed_in_space**, the nodal points of the mesh remain fixed in space in x-direction. If a *specifier_x* is set to **-follow_material**, the nodal points of the mesh will follow material displacements in x-direction. The same holds for the other directions. In 1D, you only need to give *specifier_x*, etc.

Default each specifier is set to **-follow_material**.

This record **options_mesh** only is used if **materi_velocity** is initialised. If **materi_displacement** is initialized however, each specifier is automatically set to **-fixed_in_space**.

6.311 options_nonlocal *nonlocal_radius*

By specifying this record in combination with a viscoplastic model , like **group_materi_plasti_visco_power**, a nonlocal yield rule fn will be used in the viscoplastic law. The nonlocal yield rule needs to be initialized as unknown by the **materi_plasti_f_nonlocal** record in the initialization part. The nonlocal yield rule fn in a point is determined by an averaging of the local yield rule f in neighbouring points and using gauss weighting functions for this (i.e. the larger the distance the less the neighbouring point contributes to the nonlocal yield rule). The averaging is done over a region with radius *nonlocal_radius*.

In this way, you can prevent unlimited localization and so mesh dependency, in calculations with softening plasticity.

6.312 options_processors *index nproc*

With this record you can set the number of CPUs you want to use (*nproc*). If your TOCHNOG implementation does not allow for more processors, this record is ignored. In fact, not the number of processors but the number of threads is set (that is, if you use 2 threads while your system only supports 1 processor than those threads are split over that single processor).

You can also set an environment symbol **TOCHNOG_OPTIONS_PROCESSORS** to *nproc*. In this way, you do not need to set the number of processors in each input file separately.

<u>Error messages may become confusing when you use more than one processor</u>.

Default *nproc* is 1.

6.313 options_relaxation *relax_0 relax_1 ...*

Relaxation parameters for adjusting unknowns in iterations. This can stabilize the calculation. By example, a relaxation parameter of 0.1 means that the corresponding unknown is now completely updated with the iterative change, but only 10 percent of the change is actually applied in a iteration.

If enough iterations are used, the relaxation parameters with not influence the final solution.

You should specify a relaxation parameter term for each principal unknown which is present in the calculation (see the start of the data part description for a list of principal unknowns; these are velocities, temperature, etc.).

The relaxation is used for all timesteps. See also **control_options_relaxation**.

6.314 options_residuefactor *factor_0 factor_1 ...*

While determining the residue in the differential equations **-res** for the **node_dof** records, the contribution for each primary unknown equation can be multiplied with a separate factor. Here *factor_0* is the factor for the first principal unknown, *factor_1* is the factor for the second principal unknown, etc. A factor should be specified for each principal unknown.

With these factors you can put special importance on the equation for one (or several) principal unknowns. If this record is not specified, all factors are taken to be 1.

6.315 options_skip_groundflow_materidivergence *switch*

If *switch* is set to **-yes**, then the material divergence part in the groundflow equation is skipped. This is done for all timesteps.

6.316 options_skip_plasticity *switch*

If *switch* is set to **-yes**, any plasticity data in the input file will be ignored. This is done for all timesteps.

This option is convenient for testing your input file just linear, without the need to outcomment each and every part with plasticity data. See also **control_options_skip_plasticity**.

6.317 options_solver *solver*

You can set here the solver type to one of **diagonal**, **matrix_iterative_bicg**, **matrix_iterative_petsc** or **matrix_superlu**. The solver set here holds for the entire calculation (as opposed to the **control_options_solver** which only holds for the corresponding time steps). In fact, each **control_options_solver** will be overwritten by a specified **options_solver**.

You can also set an environment symbol **TOCHNOG_OPTIONS_SOLVER** to *solver*, by example to **-matrix_superlu**. In this way, you do not need to set the solver type in each input file separately.

6.318 options_solver_petsc_ksptype *ksptype*

The same as **control_options_solver_petsc_ksptype**, but now for the entire calculation however.

6.319 options_solver_petsc_pctype *pctype*

The same as **control_options_solver_petsc_ksptype**, but now for the entire calculation however.

6.320 options_stabilization *switch*

Because of finite discretisation sometimes unrealistic results may be obtained (wiggles, etc.). If *switch* is set to **-static**, results are stabilized with a scheme which assumes that you are looking for the static (time independent) solution. If *switch* is set to **-dynamic**, results are stabilized with a scheme which assumes that you are looking for the dynamic (transient) solution. If *switch* is set to **-no**, no stabilization is used. Default *switch* is set to **-static**.

Warning: this default **-static** scheme may give bad results for transients calculations.

6.321 post_calcul *unknown operat ...*

This records activates calculation post results. Here *unknown* can be one of the matrices **-materi_stress**, **-materi_strain_elasti**, **-materi_strain_plasti**, **-materi_strain_total** or *unknown* can be one of the vectors **-materi_velocity**, **-materi_displacement**, or *unknown* can be one of the scalars **-condif_temperature**, **-groundflow_pressure**.

The results of these calculations are stored for each **node_dof** record in a **node_dof_calcul** record, and are stored for each **post_point_dof** record in a **post_point_dof_calcul** record, and are stored for each **post_line_dof** record in a **post_line_dof_calcul** record, and are stored for each **post_quadrilateral_dof** record in a **post_quadrilateral_dof_calcul** record.

We denote a matrix *unknown* with A_{ij} and denote a vector *unknown* with A_i, and denote a scalar *unknown* with a.

If *operat* is **-prival** then three principal values of a matrix A_{ij} are calculated. Each principal value contains the size of the principal vector. The principal values are ordered (the first value is the largest one, and the last value is the smallest one).

If *operat* is **-privec** then three principal vectors of a matrix A_{ij} are calculated. Each principal vector contains the x, y and z component of the principal vector. The same ordering as used for **-prival** is used here also.

If *operat* is **-average** then $\frac{1}{3}(A_{11} + A_{22} + A_{33})$ is calculated for a matrix or $\frac{1}{3}(A_1 + A_2 + A_3)$ is calculated for a vector.

If *operat* is **-absol** then the absolute value of a scalar a is calculated.

If *operat* is **-positive** then the average of the positive principal values for a matrix is calculated. If **materi_strain_plasti** is taken for the matrix A_{ij}, then this operator typically can be used as a measure for the amount of tensile failure (cracking).

If *operat* is **-negative** then the average of the negative principal values for a matrix is calculated. If **materi_strain_plasti** is taken for the matrix A_{ij}, then this operator typically can be used as a measure for the amount of compression failure (crunching).

If *operat* is **-sizetot** then $\sqrt{0.5A_{ij}A_{ij}}$ is calculated for a matrix or $\sqrt{0.5A_iA_i}$ is calculated for a vector. This measures the size of a matrix or the size of a vector.

If *operat* is **-sizedev** then $\sqrt{0.5B_{ij}B_{ij}}$ is calculated where B_{ij} is the deviatoric part of a matrix A_{ij}: $B_{ij} = A_{ij} - \delta_{ij}\frac{A_{11}+A_{22}+A_{33}}{3}$ where δ_{ij} is 1 if $i = j$ and is 0 otherwise. This measures the size of the deviatoric part of the matrix.

Specially for geotechnics you can set *operat* to **-phimob** in case *unknown* is **-materi_stress**. Then the 'mobilised friction angle' is calculated in degrees. It is default calculated without taking cohesion into account.

Specially for geotechnics you can set *operat* to **-total** in case *unknown* is **-materi_stress**. Then the total stress is calculated from the effective stress and the groundflow total pressure.

Specially for geotechnics you can set *operat* to **-static** in case *unknown* is **-groundflow_pressure**. Then the static pressure is calculated.

Specially for geotechnics you can set *operat* to **-dynamic** in case *unknown* is **-groundflow_pressure**. Then the dynamic pressure is calculated.

Specially for geotechnics you can set *operat* to **-total** in case *unknown* is **-groundflow_pressure**. Then the total pressure is calculated.

The next piece of input file

...
materi_stress
materi_strain_plasti
end_initia
...
post_calcul -materi_stress -sizedev -materi_strain_plasti -sizetot
...
control_timestep 1 ...
control_print 1 -node_dof_calcul

will print records like

node_dof_calcul *index 0.2 1.1e-4*

Here the 0.2 is the equivalent Von Mises stress and *1.1e-4* measures the plastic strain matrix.

6.322 post_error_item *index data_item_name data_item_index number*

By means of this record, you can specify a data record for which the error in its value needs to be estimated.

The error is estimated as follows. The value for the data item will be remembered just before each mesh refinement (with **control_mesh_refine_locally** or **control_mesh_refine_globally**). The last 2 stored values and the current value are used create an estimated value . The error percentage is defined as:

$$100\,\frac{actual\_value\ -\ estimated\_value}{estimated\_value}$$

The percentage will be placed in the **post_error_result**, with the same index.

A typical example looks like:

...
post_error_item 1 -post_point_dof 1 -velx
...
control_mesh_refine_globally 10 ...
...
timesteps 20 ...
...

control_repeat_until_item 30 10 -value -post_error_result 1 0
control_repeat_until_tolerance 30 5.
...

This asks for the estimation of the error in the x-velocity at a point. It is used to determine automatically the number of refinements, by requiring that the error should become less than 5 percent.

A final remark. The values used for the error estimation will not be stored if the mesh refinement takes place at the start of the calculation, since that mesh contains normally not useful information.

6.323 post_error_result *index percentage*

See **post_error_item**.

6.324 post_integrate *index data_item_name data_item_index data_item_number ...*

Here you can specify results that should be integrated over time. The integrated results will be placed in the **post_integrate_result** record with the same index.

An example looks like:

...
groundflow_pressure
groundflow_velocity
end_initia
...
post_node 1 -average -geometry_line 4
...
post_integrate 3 -post_node_result 1 -gvely
...
...

Here the **post_node** record first takes care that the average groundflow y-velocity at nodes on a line are determined, among other unknowns. The **post_integrate** record integrates that average groundflow y-velocity over time. In this way the total groundflow debit volume over a line is registered.

6.325 post_global *index name_0 name_1 ...*

With this **post_global** you can ask for global information to be determined. You can set each of the names *name_0*, *name_1*, etc. to

- **-global_elements** (total number of elements)
- **-global_nodes** (total number of nodes)

- **-global_unknowns** (total number of free principal unknowns)
- **-global_solver_iterations** (total number of iterations of iterative linear equation solver)
- **-global_mass** (total global mass)
- **-global_strainenergy** (total strain energy)
- **-global_unknown_average** (average values for unknowns)
- **-global_unknown_max** (maximum values for unknowns)
- **-global_unknown_min** (minimum values for unknowns)
- **-global_volume** (total global volume)

User **control_print** to print each of **-global_elements**, etc.

6.326 post_integrate_result *index result*

See **post_integrate**.

6.327 post_line *index x_0 y_0 z_0 x_1 y_1 z_1*

This record specifies a line in space for which the average or sum of the unknown values will be calculated. The values are placed in a record **post_line_dof** with the same *index*. Internally in TOCHNOG, **post_point** records are used to evaluate the unknowns on the line. In 1D only *x_0* and *x_1* should be specified, etc.. In the example below, the average of the x-velocity between the points (3,1) and (3,7) will be printed

```
...
number_of_space_dimensions 2
materi_velocity
...
end_data
...
post_line 1 3. 1. 3. 7.
...
print_filter 0 -post_line_dof 1 -velx
...
control_timestep 1 1. 100.
control_print 1 -post_line_dof
```

The coordinates are defined in the initial mesh. See also: **post_line_n**, **post_line_operat** and **post_line_moment**.

6.328 post_line_operat *index operat*

If *operat* is set to **-average** then the average is calculated for the **post_line** record with the same index. If *operat* is set to **-sum** then the sum is calculated for the **post_line** record with the same index. If this **post_line_operat** is not specified, then *operat* is set to **-average**.

6.329 **post_line_dof** *index dof_0 dof_1 ...*

Average unknown values at a selected line. See **post_line**.

6.330 **post_line_dof_calcul** ...

See **post_calcul**.

6.331 **post_line_moment** *index switch*

If *switch* is set to **-yes**, then the moment of the unknowns with respect to the middle of the **post_line** (same *index*) is calculated (instead of the average value of the unknowns). This moment is the integral of the unknown value multiplied with the distance to the middle of the line multiplied by the area contribution; the first point in the **post_line** has negative distance with respect to the line middle and the second point in the **post_line** has positive distance with respect to the line middle.

6.332 **post_line_n** *index n*

Use *n* **post_point** records to evaluate the unknowns along the line. Default *n* is 5. See **post_line**.

6.333 **post_node** *index data_item operat geometry_entity geometry_entity_index*

If *operat* is set to **-sum**, results for the nodal *data_item* are summed. If *operat* is set to **-average**, results for the nodal *data_item* are averaged. For example, you can take for *data_item* the **-node_rhside** data item. In this way you can sum the external nodal forces on a part of the domain.

If *operat* is set to **-moment** and *data_item* equals **node_rhside**, the moments Mx, My, Mz are determined. These are moment with respect to the three model axes, each one in the positive direction of its axis.

This operation is done for nodes which are placed on the geometrical entity *geometry_entity geometry_entity_index*. Instead of a geometrical entity you can also use **-all** to tell that all nodes should be used. The result is put into the **post_node_result** record (with the same *index*).

6.334 **post_node_result** *index result_0 result_1 ...*

See **post_node**.

6.335 **post_node_rhside_fixed** *value_0 value_1 ...*

After the calculations, this record will be filled with the absolute maxima for the **node_rhside** values for those unknowns which are fixed (so the maxima of external forces, fluxes, etc.).

6.336 **post_node_rhside_free** *value_0 value_1 ...*

After the calculations, this record will be filled with the absolute maxima for the **node_rhside** values for those unknowns which are free. In equilibrium these should become 0. Because of a limited number of equilibrium iterations however, the values may be larger then 0.

6.337 **post_node_rhside_ratio** *ratio*

After the calculations, this record will be filled with the ratio of the the internal forces (out-of-balance forces) and the external forces. The internal forces are taken from the **post_node_rhside_free** record. The external forces are taken from the **post_node_rhside_fixed** record. The maximum ratio for the unknowns is used.

As an expeption, if the external forces are very low (below 1.e-6), the *ratio* is directly filled with the internal forces and so represent an absolute value of unbalance and no unbalance ratio.

See also **post_node_rhside_ratio_unknowntypes**.

6.338 **post_node_rhside_ratio_unknowntypes** *unknowntype_0 ...*

With this option you can specify a list of unknown types which should be used in the calculation of the **post_node_rhside_ratio** result. By example, if both **materi_velocity** and **condif_temperature** are initialised, then you can should to use only the material velocities in the accuracy ratio determination by specifying **post_node_rhside_ratio_unknowntypes -materi_velocity**.

In case this **post_node_rhside_ratio_unknowntypes** is not specified, default all principal unknowns present in the calculation will be used.

6.339 **post_point** *index x y z*

This record specifies a point in space for which unknown values will be calculated. The values are placed in a record **post_point_dof** with the same *index*. The values are obtained by determining in which element the point is located and then using the element's interpolation functions. In 1D only *x* should be specified, etc.. The coordinates are defined in the initial mesh.

6.340 **post_point_dof** *index dof_0 dof_1 ...*

Unknown values at a selected point. See **post_point**.

6.341 **post_point_dof_calcul** ...

See **post_calcul**.

6.342 **post_quadrilateral** *index x_0 y_0 z_0 x_1 y_1 z_1 x_2 y_2 z_2 x_3 y_3 z_3*

This record specifies a quadrilateral in space for which the average of the unknown values will be calculated. The values are placed in a record **post_quadrilateral_dof** with the same *index*. Internally in TOCHNOG, **post_point** records are used to evaluate the unknowns on the quadrilateral.

In 2D only *x_0 y_0* and *x_1 y_1* should be specified. The coordinates are defined in the initial mesh. See also: **post_quadrilateral_n**.

6.343 post_quadrilateral_dof *index dof_0 dof_1 ...*

Average unknown values at a selected quadrilateral. See **post_quadrilateral**.

6.344 post_quadrilateral_dof_calcul ...

See **post_calcul**.

6.345 post_quadrilateral_n *index n*

Use *n* **post_point** records in each direction to evaluate the unknowns along the quadrilateral. Default *n* is 5. See **post_quadrilateral**.

6.346 print_arithmetic *switch*

If *switch* is set to **-yes**, all evaluated arithmetics will be printed. See the start of the data part for an explanation about arithmetics.

6.347 print_control *switch*

If *switch* is set to **-yes**, the control index being evaluated will be printed. Handy for keeping track on what the program is doing.

6.348 print_define *switch*

If *switch* is set to **-yes**, all evaluated defines will be printed. See the start of the data part for an explanation about defines.

6.349 print_failure *switch*

If *switch* is set to **-yes** then failure of elements due to one of the failure criteria (**group_materi_failure_rupture**, etc.) will be reported.

6.350 print_filter *index data_item_name data_item_index number_0 number_1 ...*

The data selected in the records **control_print**, **control_print_unknowns**, **control_print_unknownsrhside** or **control_print_element** will be filtered at output. Thus only a limited amount of data will actually be printed. Here *data_item_name* is the name of the data item to be filtered, e.g. *data_item_name* is **-node_dof**. *data_item_index* is the index of the *data_item_name* record which passes the filter. If, by example, *data_item_index* is 3 then only index 3 passes the filter. If *data_item_index* is **-all** then all indices pass the filter. If, by example, *data_item_index* is **-geometry_line**

3 (valid if *data_item_name* is **-node** or another nodal item) then only records with coordinates located on line 3 pass the filter. If, by example, *data_item_index* is **-geometry_line 3** (valid if *data_item_name* is **-element** or another element item) then only element with at least one coordinate located on line 3 pass the filter. If, by example, *data_item_index* is **-ra ... -ra** then indices in this range pass the filter. If, by example, *data_item_index* is **-macro 4** and *data_item_name* is data valid at a node (or element), then only nodes (or elements) generated by the macro number 4 pass the filter (see **control_mesh_macro_*** for macro's). If, by example, *data_item_index* is **-macro -none** and *data_item_name* is data valid at a node (or element) then only nodes (or elements) not generated by any macro pass the filter (see **control_mesh_macro_*** for macro's).

By example, if *number_0* is 3 then the fourth value of a record passes the filter. If *number_0* is **-all** the whole record passes the filter. If, by example, *number_0* is **-velx** while *data_item_name* is **-node_dof** then only x-velocities pass the filter.

Some examples are

> **print_filter 1 -node_dof -all -temp -sigxx** (temperatures and xx-stresses)
> **print_filter 2 -node -geometry_line 3 0** (x-coordinates on line 3)

With **control_print_filter** you can select if the records **control_print, control_print_unknowns** or **control_print_unknownsrhside** (with the same index) should use specific filters (specify the indices of the filter for *print_filter_index*), should use all filters (specify **-all** for *print_filter_index*), or should use no filter at all (specify **-none** for *print_filter_index*). Default, if **control_print_filter** is not specified, all filters will be used for a print option.

Example:

> **print_filter 1 ...**
> **print_filter 2 ...**
> ...
> **control_timestep 10 ...**
> **control_print 10 ...**
> **control_print_filter 10 1** (only use filter 1)
> ...
> **control_timestep 20 ...**
> **control_print 20 ...**
> **control_print_filter 20 -all** (use all filters)

All used filters are placed in-line; thus only data which passes all used filters will be printed.

6.351 print_lastdatabase *switch*

If *switch* is set to **-yes**, the database will be written after succesfull completion of a timestep to the file **last.dbs**.

In case of any trouble at the new timestep, that **last.dbs** can serve as a restart file. Just adjust all control records to what you want them to be.

6.352 **print_where** *switch*

If *switch* is set to **-yes**, tochnog will print some info where it is in the calculation.

6.353 **slide_geometry** *index geometry_entity geometry_entity_index*

This record makes material to slide over the geometry specified by *geometry_entity geometry_entity_index*. Velocities normal to that direction are not allowed.

This option comes handy when it is a priori known at which nodes sliding will occur, which is typically the case in an Eulerian calculation, or if deformations are small. Otherwise, contact conditions should be used (see **contact_geometry** etc.).

Optionally **slide_friction** and **slide_penalty** can be specified.

6.354 **slide_friction** *index friction_coefficient*

This specifies friction for the **slide_geometry** option.

6.355 **slide_penalty** *index penalty_coefficient*

This specifies penalty factor with which the **slide_geometry** option is enforced; if this record is not specified then some high default value for the penalty factor.

6.356 **target_item** *index data_item_name data_item_index number*

See also: **target_value**.

6.357 **target_value** *index value tolerance*

This allows for testing the results of the calculation. Typically, *data_item_name* is **-node_dof** but also other data items can be tested. The record with index *data_item_index* will be tested. If *data_item_name* is **-node_dof** then *number* can be **-velx**, **-temp**, etc. (see **dof_label**); else, by example, *number* is 3 states that the fourth value needs to be checked. The result should not differ more from *value* than *tolerance*.

For a calculation with no problems, the tn.log file contains a line stating that the calculation did start followed by a line stating that the calculation did end. If this is not precisely the case, some problem did occur or the targeted results differ too much. In the example below it is checked that the pressure in node 6 does not differ more than 1. 10^{-5}.

target_item 0 -node_dof 6 -pres
target_value 0 1.2 1.e-5

The chacked value, 1.2 in this case, has been found from a previous computation that is regarded as reliable. The present computed value is compared with the earlier one. If they agree within the specified tolerance, 1.e-5 in this case, then Tochnog is silent. If they do not, then Tochnog writes an error message into the file "tn.log".

6.358 **tendon** *index x_0 y_0 z_0 x_1 y_1 z_1 tendon_area*

This record specifies a tendon; the start point is specified by *x_0 y_0 z_0* and the end point is specified by *x_1 y_1 z_1*. The tendon has cross sectional area *tendon_area*. In 1D only *x_0*, *x_1*, etc. should be specified, etc.. Tendons will automatically be distributed over the elements. This is only available for linear and quadratic elements. Only if a tendon part completely passes an element it will be embedded in the element; tendons need to be embedded in elements which have a **-materi** differential equation **group_type**.

This record automatically fills the **element_tendon_direction**, **element_tendon_number**, **element_tendon_stress** and **element_tendon_volume** records. You are not allowed to specify those records yourself.

In combination with tendons, the **group_materi_memory** model should be set to **-total** or **-updated_without_rotation**.

6.359 **tendon_elasti** *index stiffness*

Elastic stiffness for tendon (with the same index). Tendon will use the **group_materi_memory** model of the parent element; only **-total** is available however for tendons.

6.360 **tendon_expansion** *index* α

Thermal expansion coefficient for tendon (with the same index).

6.361 **tendon_plasti** *index yield_stress*

Yield stress for tendon (with the same index). Perfectly plastic yielding is used.

6.362 **tendon_stress** *index initial_stress*

Initial stress for tendon (with the same index). See also **tendon_stress_time** and **element_tendon_stress**.

6.363 **tendon_stress_time** *index time_0 factor_0 time_1 factor_1 ...*

With this record you can specify a factor for the initial tendon stress in time (see **tendon_stress** for the initial tendon stress). A table of time points with factors should be specified. After the last time point in this table is passed, the tendon stress initially is set to the last value (that is the *initial_stress* multiplied with the last factor), and further on can change due to length changes of the tendon.

Attention: this table is only taken into account if actual time steps are performed by means of **control_timestep**.

6.364 **time_calculation** *cpu_time_in_seconds*

Consumed computer time up to moment of printing.

6.365 **time_current** *current_time*

Current time in calculation.

6.366 **volume_factor** $a_0\ a_1\ \ldots a_n$

This data item defines a polynomial in space in 1D or 2D. The polynomial specifies the cross-sectional area (in 1D) or the thickness (2D) as function of the global x coordinate (1D) or the global x,y coordinates (2D). By example, in a 1D solid calculation it can be used to specify varying cross-sectional areas of bars, or in a 1D flow calculation it can be used to specify the cross- sectional area of a channel.

In 1D the polynomial is $a_0 + a_1x + a_2x^2 + \ldots$. In 2D the polynomial is $a_0 + a_1x + a_2y + a_3x^2 + a_4xy + a_5y^2 + a_6x^3 + a_7x^2y + a_8xy^2 + a_9y^3 + \ldots$.

If this record is not specified, the cross-sectional area is 1 (1D) or the thickness is 1 (2D).

See also **volume_element_factor**.

6.367 end_data (last record of data part)

7 Runtime file

You can use a runtime file to give to Tochnog data on the fly (while it is running). The runtime file will be read at the start of each time step. Suppose the name of the normal input file is **beam.dat**, then the name of the runtime file is **beam.run**. The runtime file always needs to be ended with an **end_data** statement.

As a typical example you can use this runtime file when you are doing a long calculation and you decide while the calculation is running that you want extra output. Suppose the normal input file **tn.dat** contains:

...
control_timestep 100 ...
...

Then you can decide to get some extra GID plotting files, while Tochnog is already running, by using the runtime file **tn.run** with:

control_print_gid 100 -yes
end_data

When you want to de-activate the printing of GID files again then set the runtime file to:

control_print_gid 100 -no
end_data

As a special option, you can use **exit_tochnog -yes** in the runtime file; then Tochnog will exit the calculation after printing the complete database and GID files.

As another special option, you can use **exit_tochnog -restart** in the runtime file; then Tochnog will exit the calculation after printing the restart database and GID files (see **control_print_database** for an explanation on databases).

8 Interaction analyzes and advanced analyzes

8.1 Fluid-structure interaction

If a solid construction interacts with a fluid, both the solid and fluid can be modeled with the materi equation. Interaction forces between solid and fluid will automatically be generated. If required, a temperature field may be imposed. An example of a input file is given below

...
materi_velocity
materi_stress
condif_temperature
end_initia
...
element_group -ra -from 0 -to 100 -ra 1
element_group -ra -from 101 -to 200 -ra 2
...
type 1 -materi -condif
group_materi_elasti_young 1 ...
group_materi_memory -updated
group_condif_conductivity 3 ...
...
type 2 -materi
group_materi_elasti_compressibility 2 ...
group_materi_viscosity 2 ...
group_materi_memory -updated_without_rotation
group_condif_conductivity 2 ...
...

Elements 0-100 are solids (with temperature) and elements 101-200 are fluids (with temperature).

8.2 Consolidation analysis: ground water flow in deforming solid

The ground water flow equation can be combined with the materi equations. The solid will deform due to the ground water flow pressure gradient and ground water flow pressure will change due to solid volume changes. An example of a input file is given below

...
materi_velocity
materi_stress
groundflow_pressure
end_initia
...
groundflow_density ...
groundflow_phreaticlevel ...
...
type 0 -materi -groundflow
(solid material)
group_materi_elasti_young 0 ...

group_materi_memory -updated
(groundflow)
group_groundflow_capacity 0 . . .
. . .

The stresses as initialized by **materi_stress** are effective stresses. Internally the program calculates with total stresses (effective stress + total pressure) in the material equilibrium equation. You can obtain the total stresses for postprocessing by means of the **post_calcul** option.

To account for the gravitational stresses, use a density ρ_{sat} in the **group_materi_density** record. Here ρ_{sat} is the saturated density of the groundwater-soil mixture (mass of soil + water per unit volume of the soil-water mixture). Also specify the gravitation in the **force_gravity** record and, if required, also the **force_gravity_time** record to apply the gravitation slowly.

8.3 Heat transport in ground water flow

Heat transport in a ground water flow can be analyzed by combining the convection and diffusion of heat equation with the ground water flow equation. Now the velocity in the convection and diffusion of heat equation is taken from the groundflow velocity field ($\beta_i = {v_i}^{\mathrm{g}}$) if **groundflow_velocity** is initialized. An example of a input file is given below

. . .
groundflow_pressure
groundflow_velocity
condif_temperature
end_initia
. . .
type 0 -groundflow -condif
group_groundflow_compressibility 0 . . .
group_condif_conductivity 0 . . .
. . .

If both **materi_velocity** and **groundflow_velocity** are initialized, $\beta_i = v_i + {v_i}^{\mathrm{g}}$.

8.4 Heat transport in materials

Heat transport in a material can be analyzed by combining the convection and diffusion of heat equation with the materi equations. In this way thermal stresses or heat induced convection can be analyzed. Now the velocity in the convection and diffusion of heat equation is taken from the velocity field ($\beta_i = v_i$). An example of a input file is given below

. . .
materi_velocity
materi_stress
condif_temperature
end_initia
. . .

type 0 -materi -condif
group_materi_elasti_young 0 ...
group_materi_expansion 0 ...
group_materi_memory 0 -updated
group_condif_conductivity 0 ...
...

9 Practical tips

9.1 Continuing an analysis

- Copy the database from the previous calculation to a new file, e.g. **new.dat**.
- Adjust control records like **control_timestep**, etc.
- Run a new calculation with **new.dat**.

9.2 Saving CPU time.

- Post data items (**post_point**, etc.) slow down the calculation. Skip them if you want to spare time.
- Printing (**control_print**, etc.) can <u>substantially</u> slow down the calculation. If you know that a calculation runs fine, only print the results at the end or so.
- Redirect printed output to a file and not to the screen. A hard disk is faster then a screen.

9.3 Saving computer memory.

- Try **-options_matrix_group**.
- Try **-options_matrix_length** (for the advanced user).
- Don't use extreme large indices, since memory is allocated for all indices up to the largest.

9.4 To model geometrically linear material

- Use **-fixed_in_space** for **options_mesh**
- Use **options_convection -no**
- Use **-updated_without_rotation** for the material.

9.5 When you find yourself in times of troubles

- Try the linear elements (**-bar2**, **-tria3** and **-tet4**)!
- Try smaller time steps.
- Try **control_timestep_iterations** with at least 2.
- Try inertia (**group_condif_density**, **group_condif_capacity**, ..)
- Try relaxation (**options_relaxation**).

10 User supplied subroutines

Several skeleton user supplied subroutines are available in the file **user.cc**. You can finish a routine, compile all files and link it to get an executable. Further explanation is provided in **user.cc**.

As a special option you can use an ABAQUS **umat.f** (ABAQUS is a trademark of Hibbitt, Karlsson & Sorensen, Inc.; see http://www.abaqus.com for the ABAQUS homepage). See the **umat.c** routine in the tochnog/src directory how to do that.

See also: **user_umat**, **user_data** and **materi_history_variables**.

References

[1] H.S. Carlslaw, J.C.Jeager *1974 Conduction of heat in solids* Claredon Press Oxford

[2] Carter, J.P., 1982. Predictions of the non-homogeneous behaviour of clay in the triaxial test. *Géotechnique*, Vol. 32, No. 1, 55–58.

[3] Gudehus, G. *1996 A comprehensive constitutive equation for granular materials* Soils and Foundations, Vol. 36, No. 1, 1-12.

[4] C. di Prisco *1993 Studio sperimentale e modellazione matematica del comportamento anisotropo delle sabbie* PhD thesis in geotechnical engineering, Politecnico di Milano, Italy

[5] C. di Prisco, R. Nova and J. Lanier *1993 A mixed isotropic-kinematic hardening constitutive law for sand* Modern Approaches to Plasticity, Ed. Kolymbas, 83-124

[6] K. Morgan, J. Periaux, F. Thomasset *1984 Analysis of Laminar Flow over a Backward Facing Step* Notes on Numerical Fluid Mechanics, a GAMM-workshop Friedr. Vieweg & Sohn Braunsweig/Wiesbaden

[7] P.V. Lade and R.B. Nelson *1987 Modelling the elastic behaviour of granular materials.* Int. Jour. Num. Anal. Meth. Geomech., Vol., 2, 521-542

[8] Matsuoka, H, Nakai, T., *1977. Stress-strain relationship of soil based on the "SMP"* Proc. Specialty Session 9, IX ICSMFE, Tokyo, 153–162

[9] Matsuoka, H., Sun, D.A., Konda, T., *1994. A constitutive law from frictional to cohesive materials.* Proc. XIII ICSMFE, New Delhi, Vol. 1, 403–406

[10] Niemunis, A., Herle, I. *1997 Hypoplastic model for cohesionless soils with elastic strain range* Mechanics of Cohesive-Frictional Materials, Vol. 2, 279-299.

[11] von Wolffersdorff, P.-A. *1996 A hypoplastic relation for granular materials with a predefined limit state surface* Mechanics of Cohesive-Frictional Materials, Vol. 1, 251-271.

[12] Wood, D.M., 1990. Soil behaviour and critical state soil mechanics. Cambridge University Press.

[13] O.C. Zienkiewicz, R.L. Taylor *The finite element method Volume2, fourth edition* McGraw-Hill Book Company London

11 GNU GENERAL PUBLIC LICENSE

GNU GENERAL PUBLIC LICENSE Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the ”copyright” line and a pointer to where the full notice is found.

¡one line to give the program’s name and a brief idea of what it does.¿ Copyright (C) 19yy ¡name of author¿

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) 19yy name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’. This is free software, and you are welcome to redistribute it under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be mouse-clicks or menu items–whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a ”copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program ‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

¡signature of Ty Coon¿, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.