User's Manual Version 4.3

Last Updated: August 30, 2023

Contributors
Steve Maas (steve.maas@utah.edu)
Dr. Jeffrey Weiss (jeff.weiss@utah.edu)
Dr. Gerard Ateshian (ateshian@columbia.edu)

Contact address
Musculoskeletal Research Laboratories, University of Utah
72 S. Central Campus Drive, Room 2646
Salt Lake City, Utah

Website
MRL: http://mrl.sci.utah.edu
FEBio: http://febio.org

Forum
https://forums.febio.org/

Acknowledgments
Development of the FEBio project is supported in part by a grant from the U.S. National Institutes of Health (R01GM083925).

Contents

Chapter 1

Introduction

1.1 Overview of FEBio

FEBio is a nonlinear finite element solver that is specifically designed for biomechanical applications. It offers modeling scenarios, constitutive models and boundary conditions that are relevant to many research areas in biomechanics, thus offering a powerful tool for solving 3D problems in computational biomechanics. The software is open-source and the source code, as well as pre-compiled executables for Windows, OS-X and Linux platforms are available for download at http://febio.org. This chapter presents a brief overview of the available features of FEBio.

FEBio is a multi-physics code and can solve problems in structural mechanics, biphasic and multiphasic physics, fluid mechanics, and fluid-solid interaction (FSI). Both (quasi-) static (or steady-state) and dynamic (or transient) analyses can be performed in each of the different physics modules. For instance, in the structural mechanics module, the (quasi-) static response of the system is sought in a quasi-static analysis and the effects of inertia are ignored. In a dynamic analysis, the inertial effects are included in the governing equations to calculate the time dependent response of the system. In the biphasic module, a coupled solid-fluid problem is solved. In a transient biphasic analysis the time dependent response of both the solid and the fluid phase is determined. For the steady-state analysis the final relaxed state is recovered. Similarly, for multiphasic problems, both the time dependent transient response as well as the steady-state response can be determined. For fluid analyses, dynamic and steady-state responses may be specified.

Many nonlinear constitutive models are available, allowing the user to model the often complicated biological tissue behavior. Several isotropic constitutive models are supported such as Neo-Hookean, Mooney-Rivlin, Ogden, Arruda-Boyce and Veronda-Westmann. All these models have a nonlinear stress-strain response and are objective for large deformations. In addition to the isotropic models there are several transversely isotropic and orthotropic constitutive models available. These models exhibit anisotropic behavior in a single or multiple preferred directions and are useful for representing biological tissues such as tendons, muscles, cartilage and other tissues that contain fibers. FEBio also contains a *rigid body* constitutive model. This model can be used to represent materials or structures whose deformation is negligible compared to that of other materials in the overall model. Several constitutive models are available for representing the solid phase of biphasic and multiphasic materials, which are materials that contain both a solid phase and a fluid phase. For incompressible materials FEBio employs special algorithms for enforcing the incompressibility constraint. A three-field formulation is used for tri-linear hexahedral and wedge elements. This algorithm allows the user to capture the accurate response of highly incompressible materials.

FEBio can now also solve first-order computational homogenization problems. In such problems, the response of the macro-model is determined by the averaged local response of a representative volume element (RVE). The deformation of the macro-model, and more specifically the local deformation gradient, is applied to a RVE model which in turns determines the stress (and tangent) of the macro-model.

FEBio supports a wide range of boundary conditions and loads to model interactions between materials that are relevant to problems in biomechanics. Deformable models can be connected to rigid bodies. With this feature, the user can model prescribed rotations and torques for rigid segments, thereby allowing the coupling of rigid body mechanics with deformable continuum mechanics. FEBio provides the ability to represent frictionless and frictional contact between rigid and/or deformable materials using sliding interfaces. A sliding surface is defined between two surfaces that are allowed to separate and slide across each other but are not allowed to penetrate. Variations of the sliding interface, such as tied interfaces, tied-sliding (tension-compression) and rigid walls, are available as well. As of version 1.2 it is also possible to model the fluid flow across two contacting biphasic materials. Finally, the user may specify a body force to model the effects such as, gravity, base acceleration or centripetal acceleration.

FEBio has a large library of element formulations. These include linear and quadratic tetrahedral, hexahedral and pentahedral (wedge) elements. FEBio also supports triangular quadrilateral shell elements, with linear and quadratic interpolations.

FEBio is a nonlinear implicit FE solver and does not have mesh generation capabilities. The finite element mesh, as well as all constitutive parameters and loading is defined in an input file, the format of which is described in detail in this document. This input file needs to be generated by preprocessing software. The preferred preprocessor for FEBio is called *FEBioStudio*. FEBioStudio can convert some other formats to the FEBio input specification. For instance, NIKE3D [57] and Abaqus input files can be imported in FEBioStudio and can be exported as a FEBio input file.

1.2 About this document

This document is part of a set of three manuals that accompany FEBio: the User's Manual, describing how to use FEBio (this manual), a *Developer's Manual* for users who wish to modify or add features to the code, and a *FEBio Theory Manual*, which describes the theory behind the FEBio algorithms.[1]

This document discusses how to use FEBio and describes the input file format in detail. Chapter 2 describes how to run FEBio and explains the various command line options. It also discusses the different files that are required and created by FEBio. Chapter 3 describes the format of the FEBio input file. An XML-based format is used, organizing the data in a convenient hierarchical structure. Chapter 4 gives a detailed overview of the available constitutive models. Chapter 5 discusses the restart capabilities of FEBio. The restart feature allows the user to interrupt a run and continue it at a later time, optionally making changes to the problem data. Chapter 6 describes the multi-step analysis feature, which allows the user to split up the entire analysis into several steps. Chapter 7 explains how to setup and run a parameter optimization problem using FEBio's optimization module. Chapter 8 provides helpful information for troubleshooting an FEBio model and offers guidelines that help users avoid common problems.

[1] The User and Theory manuals are also available in pdf form, but the Developer's manual is only available online.

1.3 FEBio Basics

This section provides a brief overview of how FEBio works. For more details regarding the algorithms in FEBio, please consult the *FEBio Theory Manual*.

When FEBio starts, first it will load the configuration file where it will find instructions on what linear solver to use, what plugins to load, etc. Please see section 9 for more information regarding the configuration file.

Usually FEBio will read the input file that was specified on the command line next. The input file contains all the information that FEBio needs to build the FE Model and solve it. At the very least the input file will define the mesh, the materials, boundary conditions, time stepping, and analysis parameters. This document describes the structure of the FEBio input file in detail. See 3and 4.

Next, FEBio will start solving the model defined by the input file and proceed as follows:

1. **Loop over all analysis steps**. A model can define multiple analysis steps. In each step the boundary conditions, time stepping, and analysis parameters can be modified (e.g. a two-step analysis where a model is loaded statically in the first step. In the second step the load is released and the dynamic response is sought.)

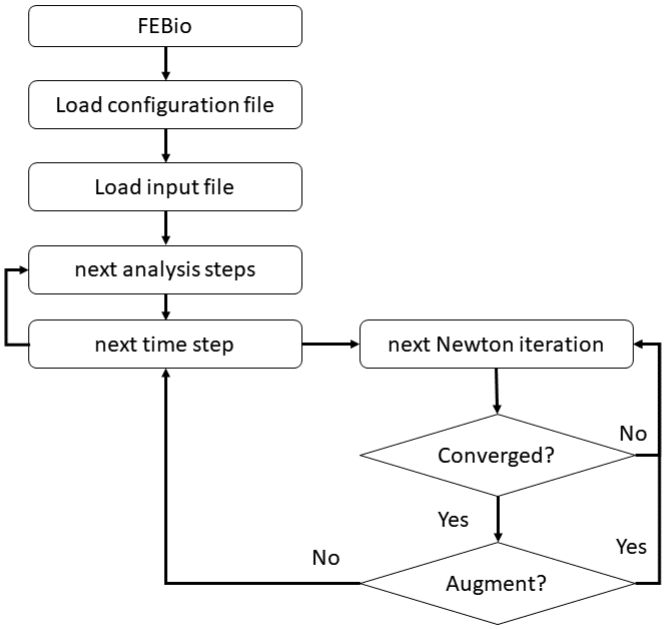
2. **For each analysis step, loop over all time steps**: In each analysis step, loads are usually applied incrementally for stability reasons over a period of time. The time parameter can represent the actual physical time (e.g. in dynamic simulations), or a pseudo-time (e.g. quasi-static loading of an elastic model.). For more on time stepping see 3.3.

3. **Solve each time step**: for each time step FEBio will solve the corresponding finite element equations. Usually they are nonlinear and thus are solved with a nonlinear solution strategy. In FEBio this is a variation of the Newton method. See section 3.3for more information.

4. **Check convergence**: Since Newton's method is an iterative method convergence is determined by comparing the norms of the solution and residual to the user-defined values in the input file. What norms are used will depend on the physics of the problem, as well as on several user control parameters. See 3.3.

5. **Augmentation**: In models that define some type of constraint (e.g. contact, rigid joints, etc.), FEBio will do an additional calculation after a time step converges, called an augmentation. This is because FEBio does not calculate the exact Lagrange multiplier that enforce the constraints, but instead uses an iterative algorithm (called augmented Lagrangian method) to approximate the Lagrange multipliers. During the augmentation, FEBio updates the approximate Lagrange multipliers. If the updates to the multipliers are small, FEBio will terminate the time step, otherwise it will restart the time step with the updated multipliers.

See the figure below for an overview of the basic FEBio flow.



1.4 Units in FEBio

FEBio does not assume a specific unit system. It is up to the user to enter numbers that are defined in consistent units. For example, when entering material parameters in SI units, the user must enter all loads, contact parameters, and other boundary conditions in SI units as well. The units of all the parameters are given when they are defined in this manual. We use a generic designation of units for all the parameters using the following symbols.

Symbol	***Name***	***SI unit***
L	Length	meter (m)
M	Mass	kilogram (kg)
t	Time	second (s)
T	Temperature	Kelvin (K)
n	Amount of substance	mole (mol)
F	Force	Newton (kg·m/s^2)
P	Pressure, stress	Pascal (Pa=N/m^2)
Q	Electric charge	Coulomb (C=A·s)

Units are given using the bracket notation. For instance, the unit for density is [**M**/**L**^{3}] and the unit for permeability is [**L**^{4}/**F**·**t**]. When using SI units, this corresponds to units of kg/m^3 for density and m^4/N.s for permeability, respectively. Unitless parameters are designated by empty brackets (**[ ]**). The units for angles are either [**deg**] for degrees or [**rad**] for radians.

When adopting a consistent set of units, first choose a primary set of units, and then determine the remaining derived units. For example, in typical problems in solid mechanics, the primary set consists of three units. If you choose [**M**]=kg, [**L**]=m, and [**t**]=s, then [**F**]=N and [**P**]=Pa. Alternatively, if you choose [**L**]=mm, [**F**]=N and [**T**]=s as the primary set, then [**P**]=MPa (since 1 N/mm^2 = 10^{6}N/m^2 = 1 MPa) and [**M**]=tonne (tonne = N ·s^2/mm). The primary set of units must be independent. For instance, in the last example, you cannot choose [**P**] as a primary unit as it can be expressed in terms of [**F**] and [**L**] (i.e. [**P**]=[**F**/**L**^{2}]).

Example:

Primary Units	
time	s
length	mm
force	N
amount of substance	nmol
charge	C
temperature	K
Derived Units	
stress	N/mm^2, MPa
permeability	mm^4/N·s, mm^2/Pa·s
diffusivity	mm^2/s
concentration	nmol/mm^3, mM
charge density	nEq/mm^3, mEq/L
voltage	mV
current density	A/mm^2
current	A

Chapter 2

Running FEBio

FEBio is a command line application which means it does not have its own Graphical User Interface (GUI) and must be run from a shell or command line. FEBio runs on several different computing platforms including Windows, Mac OSX, and many versions of Linux. The command line input and output options are described in this chapter.

2.1 Running FEBio on Windows

There are several ways to run FEBio on Windows. The easiest way is by simply selecting the FEBio program from the Programs menu or by double-clicking the FEBio icon in the installation folder. However, this runs FEBio with the installation folder as the working folder, and unless the FEBio input files are in this folder, you will need to know the relative or absolute path to your input files. A more practical approach is to run FEBio from a command prompt. Before you can do this, you need to know two things: how to open a command prompt and how to add the FEBio installation folder to your PATH environment variable so that you can run FEBio from any folder on your system.[1]

2.1.1 Windows

In order to run FEBio from a command prompt, the path to the FEBio executable must be set in the PATH environment variable. The environment variables, including the PATH variable, are specified in the Environment Variables dialog box. There are a few different ways to get to this dialog box. The easiest way is to write "environment" in the Windows search bar, and select the "Edit the system's environment variables" suggestion. This will open the "System Properties" dialog box. On the Advanced tab, click the Environment variables button. This will open the Environment Variables dialog box. Find the *path* variable and click the *Edit* button. On older versions of Windows a semicolon delimited text string appears. At the end of that string (don't delete the current value) type a semi-colon and then the absolute path to the FEBio installation folder (e.g. C:/Program Files/FEBio/bin/). On Windows 10 or newer, the values are displayed in a list. Either click New or double-click on the first empty slot in the list and type the path the febio executable. Then click the *OK*-button on all open dialog boxes.

To open a command prompt, type *cmd* in the Windows search and press Enter. A command prompt window should appear. You can now use the *cd* (change directory) command to navigate

[1] The current FEBio installers should automatically setup the environment path so that FEBio can be run from anywhere on the file system.

to the folder that contains the FEBio input files. To run FEBio, simply type *febio4* (with or without additional arguments) and press *Enter*. Note that if you already had a command prompt open when changing the PATH environment variable, you'll nee to close this window and reopen it after the changes to the environment variables are applied.

2.1.2 Running FEBio from Explorer

A third method of running FEBio, which often is very convenient, is to run FEBio from Windows Explorer. To do this, first open Explorer and browse to the folder that contains your FEBio input files. Next, right-click on the input file and select *Open With*. Now select *Choose default program*. A dialog box appears with a list of programs to open the input file. If FEBio is not on this list yet, click the *Browse* button. Locate the FEBio executable (e.g. in C:/Program Files/FEBio2/bin), select it and press the *Open* button. Now select FEBio in the *Open With* dialog box and press Ok.

After you have done this once, the process simplifies. After you right-click the input file, FEBio should now show up in the *Open With* menu item and can be selected immediately without having to go through all the previous steps.

2.2 Running FEBio on Linux or MAC

Running FEBio on Linux or Mac is as easy as opening up a shell window and typing FEBio on the command line. However, you may need to define an alias to the folder that contains the FEBio executable if you want to run FEBio from any folder on your system. Since this depends on your shell, you need to consult your Linux documentation on how to do this. E.g. if you are using c-shell, you can define an alias as follows:

```
alias febio '/path/to/febio/executable/'
```

If you don't want to define this alias every time you open a shell window, you can place it in your shell start up file (e.g. *.cshrc* for c-shell).

2.3 The Command Line

FEBio is started from a shell window (or the *command prompt* in Windows). The command line is the same for all platforms:

```
febio [-o1 [arg1] | -o2 [arg2] | ... ]
```

Where -o1, -o2 are options and *arg1*, *arg2*, ... are additional arguments. The different options (of which most are optional) are given by the following list:

A more detailed description of these options follows.

-i The -i option is used to specify the name of the input file. The input file is expected to follow the format specifications as described in Chapter 3.

Example:

```
> febio -i input.feb
```

command line option	description	argument
-i	Specify the FEBio input file.	feb file name
-r	Specify the FEBio restart file.	restart file or dump file
-g,-g1,-g2	Debug flags	-
-p	Specify name of the FEBio plot file.	plot file name
-o	Specify the name of the FEBio log file.	log file name
-s	Specify the optimization input file.	optimization input file
-d	Run an FEBio diagnostic	diagnostic input file
-break	Set a breakpoint where FEBio will pause the run.	breakpoint
-dump[=*n*]	Activate restart option and (optionally) set dump file name	dump file name
-config	Specify the FEBio configuration file to use.	configuration file
-noconfig	Don't read configuration file.	-
-nosplash	Don't print the splash window to the screen on startup.	-
-import	Import a plugin file.	plugin file name
-silent	Run FEBio in silent mode, which suppresses screen output.	-
-task[=*name*]	Run an FEBio task	task input file
-info	Print febio version information to screen or file	(optional) output file nam

Table 2.1: List of command line options.

This is the most common way to start an FEBio run. However, FEBio allows the omission of the -i when only a filename is given.

```
> febio input.feb
```

On Windows, this allows for starting FEBio by double-clicking on an input file (assuming you have chosen FEBio as the default program to open .feb files). Note that if additional options are specified on the command line the -i must be present.

It is also allowed to omit the .feb extension for FEBio input files. If no extension is given, FEBio will assume that .feb is the file extension. Thus, the following command will run the file input.feb:

```
> febio input
```

-r The -r option allows you to restart a previous analysis. The filename that must follow this option is an FEBio *restart input file* or a *dump file*. The restart input file and dump file are described in more detail in 2.8.4. The -i and -r options are mutually exclusive; only one of them may appear on the command line.

Example:

```
> febio -r file.feb
```

-g,-g1,-g2 The –g options run FEBio in *debug mode*, in which case FEBio will print more information to the log and plot files. See Section 2.8.2 for more information on running FEBio in debug mode.

Example:

```
> febio -i input.feb -g
```

-p The –p option allows the user to specify the name of the *plot file*. The plot file is a binary file that contains the main results of the analysis. FEBio usually provides a default name for this file; however, the user can override the default name using this option. See Section 2.7 for more details on the output files generated by FEBio.

Example:

```
> febio -i input.feb -p out.xplt
```

-o The –o option allows the user to set the name of the *log file*. The log file will contain a record of the screen output that was generated during a run. FEBio usually provides a default name for this file (see Section 2.7), but the user can override it with this command line option.

Example:

```
> febio -i input.feb -o out.log
```

-s This option instructs FEBio to run a material parameter optimization on the specified input file. The optimization module is described in detail in Chapter 7. The –s option is followed by the optimization control file which contains among other things the parameters that need to be optimized. Note that the restart feature does not work with the optimization module.

Example:

```
> febio -i file.feb -s control.opt
```

-d This option will run a FEBio diagnostics. A diagnostic is a special type of test that can be used to verify an implementation. For example, the *tangent diagnostic* allows users to check the consistency between the material's stress and tangent implementations.

Example:

```
> febio -d diagnostic.feb
```

-break With this option a break point can be set which sets a time point at which FEBio will interrupt the run and show the FEBio prompt. The following example sets a break point at time 1.0. FEBio will interrupt the run after the time step at time 1.0 is reached (i.e. has converged). (See section 2.8.3 for more informations on setting break points.)

Example:

```
> febio -i file.feb -break 1.0
```

-dump It is possible to restart a previous run using the restart capability in FEBio. This is useful when a run terminates unexpectedly. If that happens, the user can restart the analysis from the last converged timestep. Before this feature can be used, the user must request the creation of a *dump file*. This file will store all the information that FEBio will need to restart the analysis. FEBio will usually provide a default name for the dump file, but the *–dump* command line option allows the user to override the default name for the dump file. See Section 2.8.4 and Chapter 5 for more details on how to use the restart feature.

Example:

```
> febio -i input.feb -dump out.dmp
```

-config As of version 1.2, FEBio uses a *configuration file* to store platform specific settings. Usually FEBio assumes that the location for this configuration file is the same as the executable. However, the user can specify a different location and filename using the –config command line option. If the user does not have a configuration file or does not wish to use one, this can be specified using the –noconfig option. More details on the configuration file can be found in Section 2.5 and Chapter 9.

Example:

```
> febio -i input.feb -config C:\path\to\febio.xml
```

-noconfig Do not read and process the FEBio configuration file. FEBio will start with its default configuration.

-nosplash When the –nosplash command is entered on the command line, FEBio will not print the welcome message to the screen. This is useful when calling FEBio from another application and when the user wishes to suppress any screen output from FEBio. Other options for suppressing output can be set in the control section of the FEBio input file (see Section 3.3.1).

Example:

```
> febio -i input.feb -nosplash
```

-import This command will load the plugin that is specified following this command line option. Although it is more convienient to list plugins in the FEBio configuration file, this option allows users to load a plugin from the command line, if such a need would arise.

Example:

```
> febio -import myplugin.dll
```

-silent When the –silence option is specified on the command line, FEBio will not generate any output to the screen. Unless explicitly instructed not to, FEBio will still create a log file which will have the convergence information.

Example:

```
> febio -i input.feb -silent
```

-task FEBio will always run a *task*. The default task is to solve a forward model defined by the input file specified using the -i command line option. However, other tasks are available, such as optimization and restart, as well as user-created tasks created in FEBio plugins. The name of the task is defined after an equal sign (=). An optional task control file can be specified as well. For instance, an alternative way for running an optimization problem is as follows.

Example:

```
> febio -i input.feb -task=optimize control.opt
```

command line option	description	argument
break	Add a breakpoint, which stops FEBio at a particular point.(1)	breakpoint
breaks	Print a list of breakpoints.	-
clear	Clears one or more breakpoints.	breakpoint
config	Load (or reload) configuration file.	configuration file
cont	Continue the current task	-
conv	Force conversion of the current time step. (2)	-
debug	Toggle debug mode (3)	[on/off]
fail	Force the current time step to fail. (4)	-
help	Print a list of available commans	-
import	Import a plugin file	plugin file name
out	Output to file the current stiffness matrix and right-hand side	[-txt]
plot	Write the current model state to the plot file. (5)	-
plugins	Print a list of plugins.	-
print	Print the value of an internal variable. (6)	variable name
quit	Stop the current model or exit of no model is running	-
restart	Toggle the restart flag. (7)	-
run	Run a febio input file. (8)	(command line options
set	sets certain model and configuration variables. (9)	variable name (space) va
svg	Write the sparse matrix profile to an svg file.	svg file name
time	Print progress time statistics.	-
unload	Unload a plugin from FEBio.	plugin name
version	Print version information.	-

Table 2.2: List of options for the FEBio command prompt.

2.4 The FEBio Prompt

The FEBio prompt is shown when you start FEBio without any command arguments (referred to *interactive mode*), or when a run is interrupted either by the user (using ctrl+c [2]; See Section 2.8.1 for more details), or by reaching a breakpoint (referred to as *break mode*). The FEBio prompt will look something like this:

```
febio>
```

You can now enter one of the following commands. Note that some commands are only available in a specific mode (either *interactive* or *break* mode). Also see additional comments below.

Comments:

1. A breakpoint is a particular time point or an event at which FEBio will pause the run. See section 2.8.3 for more details on break points.
2. The *force* command is useful, for example, when a time step is having difficulty satisfying the convergence criteria. The user can then manually force the convergence of the time

[2] This feature does not work on some Linux platforms and may abruptly terminate the run.

step. However, if the convergence difficulties are due to instabilities, forcing a time step to converge could cause the solution to become unstable or even incorrect. Also be aware that even if the solution recovers on later timesteps, the manually converged step might be incorrect.

3. The *debug* command toggles the debug flag. Adding *on* (*off*) will turn the debug mode on (resp. off). In debug mode, FEBio will store additional information to the log and plot file that could be useful in debugging the run. It is important to note that since FEBio will store all non-converged states to the plot file, this file may become very large in a short number of time steps. See Section 2.8.2 for more details on debugging.

4. If the current time step is not converging and if the auto-time stepper is enabled, the fail command will stop the current time step and retry it with a smaller time step size. If the auto-time stepper is not enabled, the fail command will simply exit the application.

5. The *plot* command is useful when you want to store the non-converged state at the current iteration. Note that this command only stores the state at the current iteration. If you turn on debug mode, all the iterations are stored to the plot file.

6. The following variables can be printed.

   nnz Number of nonzeroes in global stiffness matrix.

   time The current simulation time

   neq The number of equations

7. When the restart flag is set, FEBio will create a dump file at the end of each converged time step. This dump file can then later be used to restart the analysis from the last converged time step. See Section 2.8.4 and Chapter 5 for more details on FEBio's restart feature.

8. The *run* command is used to start an FEBio task. This command takes the same options that you can enter on the command line. For example, to run a file named *test.feb* from the FEBio prompt, type the following:

   ```
   run -i test.feb
   ```

9. The following configuration file settings can be set from the febio command prompt.

   output_negative_jacobians turn on or off the detailed output when negative jacobians are encountered.

   print_model_params turn on or off the output of the parameter values of parameters that are load controlled.

   show_warnings_and_errors turn on or off the printing of warning and error messages.

   To turn the option on (off), follow the set command by the variable name and then a value of 1 (0). For example,

   ```
   set output_negative_jacobians 1
   ```

   If you enter the set command without any additional options, a list will be printed of the current configuration settings.

2.5 The Configuration File

As of version 1.2, FEBio uses a *configuration file* to store platform-specific settings, such as the default linear solver and the list of plugins that need to be loaded at startup. The configuration file uses an xml format to store data and is detailed in Chapter 9. For backward compatibility, it is still possible to run FEBio without the configuration file. In that case, the default settings prior to version 1.2 are used.

Example:

```
> febio -i myfile.feb -noconfig
```

The default configuration file needs to be stored in the same location as the executable and named *febio.xml*. Alternatively, the location and the name of the file can also be specified on the command line using the –config option.

Example:

```
> febio -i myfile.feb -config /home/my/folder/FEBio/febio.xml
```

2.6 Using Multiple Processors

As of version 2.0, FEBio uses OpenMP to parallellize several of the finite element calculations, improving the performance considerably. Both the right-hand-side and the stiffness matrix evaluations for many types of problems have been parallelized. On a system with four processors, a speedup of 2-3 can be expected, depending on the size and type of model. Models with complex material behavior (such as EFD-type materials, biphasic, multiphasic materials, etc.) will benefit most from these parallelization efforts. In addition, FEBio implements the *MKL* version of the *PARDISO* linear solver, which is a parallel linear solver that uses OpenMP.

To use multiple processors, set the environment variable OMP_NUM_THREADS to the number of desired threads. You should set the number of threads to be equal or less than the number of processors on your system (Setting it higher may actually decrease performance). For example, on a system with four processors you can set the environment as follows. On Linux using the Bash shell, execute:

```
> export OMP_NUM_THREADS=4
```

Using the c-shell, execute:

```
> setenv OMP_NUM_THREADS 4
```

Or at a Windows command prompt:

```
> set OMP_NUM_THREADS=4
```

On Windows, you can add this environment variable as well from the System Properties dialog box.

A note on repeatability Ideally, when the same model is run repeatedly, either on the same machine or on different machines, it should produce the same convergence statistics and results. However, in practice this is not always the case and sometimes the convergence stats and even the results can differ. In most cases, the discrepancies should be small, however in some cases, and especially in models that are prone to ill-conditioning (e.g. contact), the discrepancies may be more significant. The underlying reason is that in different compute environments it can not be guaranteed that all calculations are executed in the exact same order and, due to numerical round-off, the results of these calculations will not always be the same. On a single machine, this can happen when the model is run using multiple processors. (However, running the same model on a machine with a single processor should always produce the exact same results.) This behavior can also be observed when running the same model on different machines, especially if these machines have different OS.

Similarly, although FEBio aims to be backward compatible, running the same model with different versions of FEBio, may also show discrepancies for similar reasons. In this case, the discrepancies are likely caused by algorithmic optimizations or changes in compiler settings.

2.7 FEBio Output

2.7.1 Screen output

By default, FEBio will print convergence and progress information to the screen that informs users how the analysis is progressing. The output depends on the particular module and solver that was chosen in the FEBio input file, as well as some user-defined settings, but in general provides the following information.

- **Time stepping information**: The current time step that FEBio is solving and a notification when the time step completed (or an error message why the time step failed to complete)
- **Convergence information**: for each time step, FEBio usually has to solve a nonlinear problem for which an iterative nonlinear solver is used. Several “norms” are printed to inform the user how FEBio is progressing toward the solution of the nonlinear problem.
- **Summary:** At the end of the analysis a summary is printed with heuristics that inform the user how well the model ran.

By default, all screen output will also be printed to the log flie.

2.7.2 Output files

After running FEBio, two or three files are created: the *log file*, the *plot file,* and optionally the *dump file*. The log file is a text file that contains the same output (and usually more) that was written to the screen. The *plot file* contains the results of the analysis. Since this is a binary file, the results must be analyzed using post processing software such as *FEBio Studio*. In some cases, the user may wish to request the creation of a *dump file*. This file contains temporary results of the run. If an analysis terminates unexpectedly or with an error, this file can be used to restart the analysis from the last converged time step. See Section 2.8.4 and Chapter 5 for more details. The names of these files can be specified with the command options –p (plot file), -a (dump file), -o (log file). If one or more of the file names following these flags are omitted, then the omitted file name(s) will be given a default name. The default file names are derived from the input file name. For

example, if the input file name is *input.feb* the logfile will have the name *input.log*, the plot file is called *input.xplt* and the dump file is called *input.dmp*.

Note 1. The name of the log and plot file can also be specified in the FEBio input file. See Section 3.18 for more information.

Note 2. When running an optimization problem the name of the log file is derived from the optimization control file. See Chapter 7 for more information on running optimization problems with FEBio.

2.8 Advanced Options

2.8.1 Interrupting a Run

The user can pause the run by pressing ctrl+c. This will bring up the FEBio prompt, and the user can enter a command. The FEBio prompt will also be shown when FEBio reaches a break point. See Section 2.4 for a list of available commands. Note that it may take a while before the FEBio prompt is displayed after the user requests a ctrl+c interruption. This may be because the program is in the middle of a call to the linear solver or another time-consuming part of the analysis procedure that cannot be interrupted. NOTE: This feature may not work properly on all systems, although it should always work on Windows systems.

2.8.2 Debugging a Run

As stated in Section 2.3, FEBio can be run in debug-mode by specifying the –g option on the command line. When running in debug mode, FEBio performs additional checks and prints out more information to the screen and to the plot file. It will also store all non-converged states to the plot file. These non-converged states can be very useful for determining the cause of non-convergence or slow convergence. Because of this additional work, the problem may run slightly slower. Note that debug mode can be turned on/off while running an analysis by first interrupting the run with ctrl+c and then using the *debug* command to toggle the debug mode on or off. It is important to note that since FEBio will store all non-converged states to the plot file, this file may become very large in a short number of time steps. An alternative approach is to use the *plot* command to write out select non-converged states.

2.8.3 Setting break points

Although ctrl+c can be used to interrupt a run, it might sometimes be difficult to interrupt a run at a desired point. For this reason, you can set break points that will pause the execution and bring up the FEBio prompt. Breakpoints are set on the command line using the **-break** option, followed by the break condition. In addition, breakpoints can also be set via the FEBio prompt using the **break** option.

To break at a particular time point, simply add the value of the time after the *-break* option. This example will pause the run after time 0.45 completed (or the first time point past 0.45):

```
>febio -i input.feb -break 0.45
```

Instead of pausing at a particular time, you can also interrupt the run when a certain event happens. To break at an event, specify the event name at which to pause the run. The following list of events are defined.

Event	Description
ALWAYS	break on any event
INIT	break after model initialization
STEP_ACTIVE	break after step activation
MAJOR_ITERS	break after major iteration (i.e. time step) converged
MINOR_ITERS	break after minor iteration (i.e. Newton iteration)
SOLVED	break after the model is solved
UPDATE_TIME	break before time is incremented
AUGMENT	break before augmentation
STEP_SOLVED	break after step is solved
REFORM	break after global matrix and right hand side is reformed
MATRIX_SOLVE	break right before the linear system solve

For example, to pause the execution after a matrix reformation (and right-hand-side evaluation), enter the following command line.

```
>febio -i input.feb -break REFORM
```

Once the code reaches a breakpoint, the febio prompt will be presented. There are several commands that relate to breakpoints.

break Add a breakpoint

breaks Prints list of current breakpoints.

clear Clear one or all breakpoints.

2.8.4 Restarting a Run

When the creation of a restart dump file is requested, the analysis can be restarted from the last converged timestep. This is useful when the run terminated unexpectedly or when the user wishes to modify some parameters during the analysis. To request the creation of a dump file, simply add the *-dump* flag on the command line. This will generate a dump file which then can be used to restart the analysis. You can specify an optional dump level, which sets how often the dump file is created, and an optional dump file name.

```
> febio -i input.feb -dump
```

With the optional arguments, the complete syntax would be:

```
>febio -i input.feb -dump=1 out.dmp
```

The dump level can be set to 1 (= write dump file after each converged time step), 2 (= write dump file after each completed analysis step), or 3 (= write dump file after each converged *must-point*).

To restart an analysis, use the -r command line option. This option requires a filename as a parameter, and this name can be either the name of a dump file or the name of a restart input file. The latter case is a text file that allows the user to redefine some parameters when restarting the run. The format of this file is described in Chapter 5.

2.9 FEBio File Conventions

Although users are free to choose file names and file extensions, it might be beneficial to adhere to the following file naming conventions.

feb FEBio input file.

xplt FEBio plot file.

log FEBio log file.

dmp FEBio dump file (for restarts).

opt FEBio optimization control file.

rst FEBio restart input file.

Chapter 3

Free Format Input

This chapter describes the XML-based input format used by FEBio. Since this format follows standard XML conventions, the files can be viewed with any file viewer that supports XML files. Since the free format input file is a text file, it can be edited with any text editor.

An XML file is composed of a hierarchical list of *elements*. The first element is called the *root element*. Elements can have multiple *child elements*. All elements are enclosed by two *tags:* a tag defining the element and an *end tag*. A simple example of an XML file might look like this:

```
<root>
  <child>
    <subchild> ... </subchild>
  </child>
</root>
```

The *value* of an element is enclosed between the name and the end tag.

```
<element> here is the value </element>
```

Note that the XML format is case-sensitive.

XML elements can also have *attributes* in name/value pairs. The attribute value must always be quoted using quotation marks (") or apostrophes (')[1].

```
<element attr="value">...</element>
<element attr='value'>...</element>
```

If an XML element has no value, an abbreviated syntax can be used. The following two lines are identical.

```
<element [attribute list]></element>
```

or

```
<element [attribute list]/>
```

Comments can be added as follows.

[1] Support for apostrophes was not added until FEBio version 2.1.

```
<!-- This is a comment -->
```

The first line in the document – the XML declaration – defines the XML version and the character encoding used in the document. An example can be:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

3.1 Free Format Overview

The free format organizes the FEBio input data into hierarchical XML elements. The root element is called *febio_spec*. This root element also defines the format version number (Note that FEBio and the input format specification follow different version numberings). This document describes version 4.0 of the FEBio specification[2] (see Section 3.1.1 below for more details on the different input specification formats). The root element will therefore be defined as follows:

```
<febio_spec version="4.0">
<!-- contents of file -->
</febio_spec>
```

The different sections introduced in this chapter are child elements of this root element. The following sections are currently defined:

Module defines the physics module for solving the model.

Globals Defines the global variables in the model

Control specifies control and solver parameters.

Material Specifies the materials used in the problem and the material parameters.

Mesh Defines the mesh of the problem, including nodal coordinates and element connectivity.

MeshDomains Assigns materials and other formulation attributes to element sets.

MeshData Defines element, facet, edge or nodal data that can be mapped to material parameters or certain boundary conditions and loads.

MeshAdaptor Defines mesh adaptors that modify the mesh.

Initial Defines initial conditions for dynamic problems, such as initial velocities, and for transient quasi-static problems.

Boundary Defines the boundary conditions that are applied on the geometry.

Loads Defines the loads applied to the model. This includes nodal loads, boundary loads and volume loads (or sources for heat transfer problems).

Contact This section defines all contact interfaces.

Constraints This section defines rigid and nonlinear constraints.

[2] FEBio continues to read some older formats, but they are considered to be obsolete.

Rigid This sections defines rigid components, such as rigid constraints, rigid loads, and rigid connectors.

Discrete This section defines all the discrete elements (i.e. springs).

LoadData Defines the load controllers, which can control model parameters as a function of time.

Step Defines different analysis steps, where in each analysis the boundary, loads, contact and initial conditions can be redefined.

Ouput Defines additional data that needs to be stored to file.

Although there is some flexibility in the order in which these sections can be listed, it is recommended to list them in the same order as given above. Not all sections are required. Empty sections can be omitted. A minimal file must contain at least the Module, Control, Material, Mesh, and MeshDomains sections. The rest of this chapter describes each of these sections in more detail.

3.1.1 Format Specification Versions

This document describes version 4.0 of the FEBio input specification. This format differs in several aspects from the previous versions of the input specification. This section describes the major changes between the different versions.

- **Version 4.0**: The latest and recommended version of the FEBio input specification described in this document. The main motivation for this format was to improve consistency between the feb file structure and the way the FEBio model is constructed in FEBio Studio. This format is very similar to version 3.0 and only differs in a few sections: Some solver parameters have moved. The Rigid section was restructured. Node and element sets can be specified more concisely.
- **Version 3.0**: The major focus of this revision was adding support for FEBio 3.0's features for modeling heterogeneous parameters. Heterogeneous parameters can be created via mathematical expressions or via the MeshData section. It also introduces a more consistent way for referencing model parameters. The *Geometry* section was replaced with the *Mesh* and *MeshDomains* sections in order to further separate the definition of the mesh and its physical attributes (e.g. material assignments). This format is **still supported** but considered obsolete.
- **Version 2.5**: This format differs from its predecessor in some important aspects: all nodesets, surfaces, etc., that are used by boundary conditions, loads, contact, etc., must be defined in the Geometry section. Boundary conditions, loads, contact, etc., are now defined by referencing the sets in the Geometry section. This format also adds the *MeshData* section and reformats the *Discrete* section. Rigid node sets and prescribed rigid degrees of freedom are moved to the *Boundary* section. This format is **still supported** but considered obsolete.
- **Version 2.0**: This is the first major revision of the input file format and redefines many of the file sections: The *Elements* section uses a different organization. Elements are now grouped by material and element type. Multiple *Elements* sections can now be defined to create multiple parts. Surfaces can now be defined in the *Geometry* section and referenced by boundary conditions and contact definitions. A new *Contact* section contains all the

contact definitions. A new *Discrete* section was defined that contains all the materials and definitions of the discrete elements (e.g. springs). The *Boundary* section is also redesigned. This format is **still supported** but considered obsolete.

- **Version 1.3**: This was an experimental version that redefined the *Geometry* section, but was later abandoned in favor of version 2.0. This version is **no longer supported**.
- **Version 1.2**: A *Loads* section was added and all surface and body loads are now defined in this section instead of the *Boundary* section. This version is **mostly supported** but considered obsolete.
- **Version 1.1**: Rigid body constraints are no longer defined in the rigid material definition but instead placed in a new *Constraints* section. This version is **no longer supported**.
- **Version 1.0**: The original input format specification. This version is **no longer supported**.

As of FEBio 4.0, only versions 1.2, 2.0, 2.5, and 3.0 are supported. Versions 1.2 and 2.0 are considered obsolete and it is highly recommended to convert older files to the newest specification for use with newer versions of FEBio. This can be done for instance using FEBioStudio.

3.1.2 Notes on backward compatibility

Some features that were available in versions prior to 4.0 are no longer supported or require a different syntax.

1. The *Parameters* section is no longer supported. This section allowed users to define file parameters that could be used as parameter values. This section was removed since it was difficult to support in FEBio Studio and a better mechanism was implemented for defining model-level parameters. (See the *3.4.4* section.)
2. Some fiber materials defined the *theta* and *phi* parameters for setting the fiber orientation. These parameters are no longer supported. Instead, the fiber direction should be specified via the *fiber* property, or if the *fiber* property is not available, the *mat_axis* property.

```
<fiber type="angles">
  <theta>90</theta>
  <phi>0</phi>
</fiber>
```

3.1.3 Multiple Input Files

FEBio supports distributing the model definition across multiple input files. This can greatly facilitate defining large, complex models and allows the re-use of model input files without the need to create the entire model input file from scratch. When using multiple input files to define a model, you must create a control input file that may reference other input files. This control file will be used to run the model in FEBio. There are two ways of including a file into the control file: The *Include* section, and the *from* attribute.

Note that all input files must start with the *febio_spec* tag and use the same format specification.

3.1.3.1 Include Keyword

The *Include* keyword[3] can be used to include the contents of another FEBio input file. The filename is entered as the value of the tag.

```
<Include>example.feb</Include>
```

The included file must be a valid FEBio file in that it must begin with the *febio_spec* tag and contain sections defined in this document. However, the included file does not need to define a complete model definition. For instance, it can contain only the *Mesh* section.

Note that the contents of the entire file will be included. This is different from the *from* attribute discussed below, which can be used to include only certain sections from files.

3.1.3.2 The 'from' Attribute

The *from* attribute can be used to include sections from other files. All the main sections defined in Section 3.1 support the *from* attribute which can be used to load the section from another input file. For example, to load the *Material* section from the file mat.feb, defining the *Material* section in the control input file as follows.

```
<Material from="mat.feb"/>
```

FEBio will now read the *Material* section from this child file. The child file must be a valid FEBio input file, meaning it must begin with the *febio_spec* root section, but does not have to be complete. For example, the file mat.feb only needs to define the Material section. However, the child file may contain other sections. In that case, only the section referenced in the control file will be read from the child file. For example, if the file *in.feb* contains both the *Material* and the *Mesh* section, the control file can read both these sections as follows.

```
<Material from="in.feb"/>
<Mesh from="in.feb"/>
```

To give a more concrete example, assume that the *Material*, *Mesh*, and *Boundary* sections are defined in the files *mat.feb*, *geom.feb*, and *bc.feb* respectively. The control input file could then look like the following.

```
<febio_spec version="4.0">
  <Module type="solid"/>
  <Control>
    <time_steps>10</time_steps>
    <step_size>0.1</step_size>
  </Control>
  <Material from="mat.feb"/>
  <Mesh from="geom.feb"/>
  <Boundary from="bc.feb"/>
</febio_spec>
```

[3] Supported from FEBio version 2.3 and up.

Notice that the *Control* section is still defined in the control file. The control file can contain a combination of explicit section definitions and referenced sections using the *from* attribute. As mentioned above, the control file is used to run the model in FEBio. So, if the control file is called *model.feb* then the model is run as follows.

```
>febio -i model.feb
```

When FEBio parses the control file it will automatically parse the referenced child files it encounters in the control input file.

3.2 Module Section

The module section defines the type of analysis to perform with FEBio. This section must be defined as the first section in the input file. It takes on the following format:

```
<Module type="[type]"/>
```

where *type* can be any of the following values:

type	Description
solid	Structural mechanics analysis: quasi-static or dynamic, implicit and explicit solver available
biphasic	Biphasic analysis: steady-state or transient
solute	Biphasic analysis including solute transport: steady-state or transient
multiphasic	Multiphasic analysis including solute transport and chemical reactions
heat	Heat transfer analysis: steady-state or transient (1)
fluid	Fluid mechanics analysis: steady-state or dynamic
fluid-FSI	Fluid mechanics with fluid-structure interactions: steady-state or dynamic

The module tag can also be used to specify the units of the model. Although FEBio doesn't care about units, if the user specifies the unit system, the proper units of variables are stored in the plot file and visible in FEBio Studio when opening the plot file.

To specify units, use the following syntax:

```
<Module type="[type]">
  <units>[unit system]</units>
</Module>
```

where *unit system* can be any of the following values.

units	Description
SI	Units follow the standard definition of the international SI unit system.
mm-N-s	Millimeter-Newton-seconds unit system
mm-kg-s	Millimeter-kilogram-seconds unit system
um-nN-s	Micron-nanoNewton-seconds unit system
CGS	Centimeter-gram-seconds unit system

If the units are not defined, FEBio will not output any units for plot variables.

For example:

```
<febio_spec version="4.0">
  <Module type="solid"/>
  <!-- rest of file -->
</febio_spec>
```

The following example defines the model's units explicitly.

```
<febio_spec version="4.0">
  <Module type="solid">
    <units>SI</units>
  </Module>
  <!-- rest of file -->
</febio_spec>
```

Comments:

1. The heat module requires the FEBioHeat plugin.

Notes:

1. In version 1.2 the Module section was optional. If omitted it was assumed that the *solid* module was used. Since version 2.0 the Module section is required and must be the first section in the file.
2. Older versions of FEBio (format specification 1.2 and before) allowed you to run a poroelastic (now called biphasic) problem by simply defining a poroelastic material. This is no longer possible. You need to define the proper Module section to run a biphasic analysis. If you have a file that no longer works as of version 1.4 of FEBio, you'll need to insert the following Module section in the file as the first section of the file.

```
<febio_spec version="1.2">
  <Module type="biphasic"/>
  <!-- rest of the file unaltered -->
</febio_spec>
```

3.3 Control Section

The control section is defined by the *Control* element. This section defines all parameters that are used to control the evolution of the solution as well as parameters for the nonlinear solution procedure. These parameters are defined as child elements of the *Control* element and depend somewhat on the analysis as defined by the *Module* section. The control section defines the control parameters, the solver parameters, and optionally, the time_stepper parameters.

Example:

```
<Control>
    <analysis>STATIC</analysis>
    <time_steps>50</time_steps>
    <step_size>0.02</step_size>
    <solver>
        <max_refs>25</max_refs>
        <diverge_reform>1</diverge_reform>
        <reform_each_time_step>1</reform_each_time_step>
        <dtol>0.001</dtol>
        <etol>0.01</etol>
        <rtol>0</rtol>
        <lstol>0.9</lstol>
        <min_residual>1e-20</min_residual>
        <qn_method type="BFGS"/>
            <max_ups>10</max_ups>
        <rhoi>-2</rhoi>
    </solver>
    <time_stepper>
        <dtmin>0.0002</dtmin>
        <dtmax>0.02</dtmax>
        <max_retries>5</max_retries>
        <opt_iter>6</opt_iter>
    </time_stepper>
</Control>
```

3.3.1 Control Parameters

The following parameters are common for all analysis types. If not specified they are assigned default values, which are found in the last column. An asterisk (*) after the name indicates a required parameter. The numbers behind the description refer to the comments following the table.

Parameter	Description	Default
analysis	Sets the analysis type (1)	static
time_steps*	Total number of time steps. (= *ntime*)(2)	(none)
step_size*	The initial time step size. (= *dt*) (2)	(none)

plot_level	Sets the level of state dumps to the plot file (3)	PLOT_MAJOR_ITRS
plot_range	Set the range of the states that will be stored to the plot file (4)	0,-1
plot_stride	Set the stride of the states that will be stored to the plot file (4)	1
plot_zero_state	Flag that controls whether the "zero" state will be written to the plot file, even if it is not defined in the range (4)	0 (false)
print_level	Sets the amount of output that is generated on screen (5)	PRINT_MINOR_ITRS
output_level	Controls when to output data to file (6)	OUTPUT_MAJOR_ITRS

Comments:

1. The *analysis* element sets the analysis type. The text value of this element is tdetermined by the module, but the numeric value should be 0 for (quasi-)static or steady-state analysis and 1 for dynamic or transient analysis. In a quasi-static analysis, inertial effects are ignored and an equilibrium solution is sought. Note that in this analysis mode it is still possible to simulate time dependant effects such as viscoelasticity. In a dynamic analysis the inertial effects are included.

Value	**Description**
static	(quasi-) static analysis
steady-state	steady-state response of a transient (quasi-static) biphasic, biphasic-solute, multiphasic, fluid or fluid-FSI analysis
dynamic	dynamic analysis for solid, fluid and fluid-FSI analyses

2. The total simulation time of the analysis is determined by *ntime* * *dt*. Note that when the auto-time stepper is enabled (see below), the actual number of time steps and time step size may be different than specified in the input file. However, the total simulation time will always be determined by *ntime* * *dt.*

3. The *plot_level* allows the user to control exactly when the solution is to be saved to the plot file. The following values are allowed:

Value	**Description**
PLOT_NEVER	Don't save the solution
PLOT_MAJOR_ITRS	Save the solution after each converged timestep
PLOT_MINOR_ITRS	Save the solution for every quasi-Newton iteration
PLOT_MUST_POINTS	Only save the solution at the must points

The PLOT_MUST_POINTS option must be used in conjunction of a *must-point* curve. See the comments on the *dtmax* parameter for more information on must-point curves. When the *plot_level* option is set to PLOT_MUST_POINTS, only the time-points defined in the must-point curve are stored to the plotfile.

4. When using the fixed time stepper, several parameters control which time steps are stored to the plot file.

   The *plot_range* parameter sets the range of the states that will be stored to the plot file. The range is defined by two values that specify the first and last time step that will be stored to the plot file. The value "0" refers to the initial time step, usually time zero, and negative values count backwards from the final time step (as defined by the *time_steps* parameter). For instance, the default values,

   ```
   <plot_range>0,-1</plot_range>
   ```

   store all time steps to the plot file, including the initial "zero" time step. As another example, to store only the last five time steps, set

   ```
   <plot_range>-5,-1</plot_range>
   ```

   By default, all time steps within the range will be stored to the plot file. Time steps can be skipped using the *plot_stride* parameter. For instance, to store only every 10 steps, set

   ```
   <plot_stride>10</plot_stride>
   ```

   Note that the first and last time step defined by the range will always be stored, regardless of the plot stride.

   The "zero" time step refers to the initial state of the model, before any calculations are done. This state will only be stored to the plot file if the minimal value of the plot range is set to zero. To force storing this state to the plot file, set the *plot_zero_state* parameter to one.

   ```
   <plot_zero_state>1</plot_zero_state>
   ```

   This will store the zero time step to the plot file, even when it is not specified inside the plot range.

   Again, the *plot_range*, *plot_stride*, and *plot_zero_state* parameters are only used by the fixed time stepper. Currently, these parameters are not used with the auto time stepper.

5. The *print_level* allows the user to control exactly how much output is written to the screen. The following values are allowed:

Value	**Description**
PRINT_NEVER	Don't generate any output
PRINT_PROGRESS	Only print a progress bar
PRINT_MAJOR_ITRS	Only print the converged solution
PRINT_MINOR_ITRS	Print convergence information during equilibrium iterations
PRINT_MINOR_ITRS_EXP	Print additional convergence info during equilib. iterations

6. The *output_level* can be used to control when FEBio outputs the data files. The following values are supported.

Value	Description
OUTPUT_NEVER	Don't generate any output
OUTPUT_MUST_POINTS	Only output at must points
OUTPUT_MAJOR_ITRS	Output at end of each time step
OUTPUT_MINOR_ITRS	Output at each iteration
OUTPUT_FINAL	Only output the data at the last converged time step.

3.3.2 Time Stepper parameters

The optional *time_stepper* element sets the parameters that control the FEBio auto-time stepper. This auto-time stepper will adjust the time step size depending on the convergence stats of the previous time step. It defines the following parameters.

Value	Description	Default
dtmin	Minimum time step size (1)	
dtmax	Maximum time step size (3)	
opt_iter	Optimal, or desired, number of iterations per time step (2)	10
max_retries	Maximum number of times a time step is restarted. (2)	5
aggressiveness	Algorithm for cutting the time step size after a failed time step (4)	0
cutback	Time step scale factor, used when aggressiveness = 1	0.5

Comments:

1. The *dtmin* and *dtmax* values are used to constrain the range of possible time step values. The *opt_iter* defines the estimated optimal number of quasi-Newton iterations. If the actual number of iterations is less than or equal to this value the time step size is increased, otherwise it is decreased.

2. When a time step fails (e.g. due to a negative jacobian), FEBio will retry the time step with a smaller time step size. The *max_retries* parameter determines the maximum number of times a timestep may be retried before FEBio error terminates. The new time step size is determined by the ratio of the previous time step size and *max_retries+1*. For example, if the last time step size is 0.1 and *max_retries* is set to 4, then the time step size is adjusted by 0.02: The first retry will have a step size of 0.08; the next will be 0.06, and so on.

3. The user can specify a load curve for the *dtmax* parameter. This load curve is referred to as the *must-point* curve and serves two purposes. Firstly, it defines the value of the *dtmax* parameter as a function of time. This can be useful, for instance, to enforce smaller time steps during rapid loading or larger time steps when approaching steady-state in a transient analysis. Secondly, the time points of the *dtmax* loadcurve define so-called *must-points*. A must-point is a time point where FEBio must pass through. This is useful for synchronizing the auto-time stepper with the loading scenario. For instance, when loading starts at time 0.5, adding a must-point at this time will guarantee that the timestepper evaluates the model at that time. In conjunction with the PLOT_MUST_POINT value of the *plot_level* parameter, this option can also be used to only write results to the plotfile at the specified time points. Consider the following example.

```
<dtmax lc="1">0.1</dtmax>
...
<loadcontroller id="1" type="loadcurve">
  <interpolate>STEP</interpolate>
    <points>
      <point>0,0</point>
      <point>0.5, 0.1</point>
      <point>1.0, 0.2</point>
    </points>
</loadcontroller>
```

This example defines load curve 1 as the *must-point* curve. This curve defines three points where FEBio will pass through (namely 0, 0.5 and 1.0). The values of each time point is the value of the maximum time-step size (*dtmax*). Since the curve is defined as a step-function, each value is valid up to the corresponding time-point. Thus, between time 0 and time 0.5, the maximum time step value is 0.1. Between 0.5 and 1.0 the maximum time step value is 0.2. If the *plot_level* parameter is set to PLOT_MUST_POINTS, then only the three defined time points will be stored to the plotfile.

4. When FEBio fails to converge a time step, the time step size is reduced and then FEBio tries to solve the time step again. The algorithm for determining the reduced time step size, depends on the *aggressiveness* parameter.

   (a) *aggressiveness = 0* (default): $\Delta t_{k+1} = \Delta t_k - (\Delta t_k/m_r)$, where $m_r$is the max numer of retries (set by the *max_retries* parameter).

   (b) *aggressiveness = 1:* $\Delta t_{k+1} = \gamma \Delta t_k$, where γ is the cutback factor (i.e. the *cutback* parameter).

3.3.3 Common Solver Parameters

Many of the solvers define the same parameters. They are listed in the table below. Additional parameters that are only defined for a specific type of analysis are listed in the following sections.

Parameter	Description	Default
etol	Convergence tolerance on energy (1)	0.01
rtol	Convergence tolerance on residual (1)	0 (disabled)
lstol	Convergence tolerance on line search (2)	0.9
lsmin	minimum line search step	0.01
lsiter	Maximum number of line search iterations	5
max_refs	Max number of stiffness reformations (3)	15
qn_method	Quasi-Newton update method (3)	BFGS
max_ups	Max number of BFGS/Broyden stiffness updates (3)	10
cmax	Set the max condition number for the stiffness matrix (4)	1e+5

diverge_reform	Flag for reforming stiffness matrix when the solution diverges (3)	1
min_residual	Sets minimal value for residual tolerance (6)	1e-20
symmetric_stiffness	Use symmetric stiffness matrix flag (7)	1
equation_scheme	Set the equation allocation scheme	0
equation_order	Set the equation allocation ordering	0
optimize_bw	Optimize bandwidth of stiffness matrix (5)	0
linear_solver	Set the preferred linear solver to use. (8)	(default)

Comments:

1. FEBio determines convergence of a time step based on three convergence criteria: displacement, residual and energy (that is, residual multiplied by displacement). Each of these criteria requires a tolerance value that will determine convergence when the relative change will drop below this value. For example, a displacement tolerance of ε means that the ratio of the displacement increment (i.e. the solution of the linearized FE equations, $\Delta\mathbf{U} = -\mathbf{K}_k^{-1}\mathbf{R}_k$) norm at the current iteration $k+1$ to the norm of the total displacement ($\mathbf{U}_{k+1} = \mathbf{U}_k + \Delta\mathbf{U}$) must be less than ε:
$$\frac{\|\Delta\mathbf{U}\|}{\|\mathbf{U}_{k+1}\|} < \varepsilon$$
For the residual and energy norms, it is the ratio of the current residual norm (resp. energy norm) to the initial one that needs to be less than the specified convergence tolerance.

   To disable a specific convergence criterion, set the corresponding tolerance to zero. For example, by default, the residual tolerance is zero, so that this convergence criterion is not used.

2. The *lstol* parameter controls the scaling of the vector direction obtained from the line search. A line search method is used to improve the convergence of the nonlinear (quasi-) Newton solution algorithm. After each BFGS or Newton iteration, this algorithm searches in the direction of the displacement increment for a solution that has less energy (that is, residual multiplied with the displacement increment) than the previous iteration. In many problems this will automatically be the case. However, in some problems that are very nonlinear (e.g. contact), the line search can improve convergence significantly. The line search can be disabled by setting the *lstol* parameter to zero, although this is not recommended.

3. The *qn_method* property determines the quasi-Newton update method. The particular method is set via the *type* attribute. Additional parameters are then specified as child elements of this tag. See section 3.3.4 for more details.

4. When the condition number of the stiffness matrix becomes too large, inversions of the stiffness matrix become inaccurate. This will negatively affect the convergence of the quasi-Newton or Newton solution algorithm. FEBio monitors the condition number of the BFGS stiffness update and when it exceeds *cmax* it reforms the stiffness matrix.

```
<cmax>1e5</cmax>
```

5. The *optimize_bw* parameter enables bandwidth minimization for the global stiffness matrix. This can drastically decrease the memory requirements and running times when using the skyline solver. It is highly recommended when using the skyline solver.

```
<optimize_bw>1</optimize_bw>
```

   When using a different linear solver (e.g., pardiso or SuperLU), the bandwidth optimization can still be performed if so desired. However, for these solvers there will be little or no effect since these solvers are not as sensitive to the bandwidth as the skyline solver.

6. If no force is acting on the model, then convergence might be problematic due to numerical noise in the calculations. For example, this can happen in a displacement driven contact problem where one of the contacting bodies is moved before initial contact is made. When this happens, the residual norm will be very small. When it drops below the tolerance set by *min_residual*, FEBio will assume that there is no force acting on the system and will converge the time step.

7. The *symmetric_stiffness* flag is used to set the symmetry of the global stiffness matrix. Only specify this flag if you want to override the default symmetry, which depends somewhat on the analysis type. Forcing a symmetric matrix when the problem is non-symmetric may have negative impact on convergence. The values for this parameter are:
   - 0 = unsymmetic
   - 1 = symmetric
   - 2 = structurally symmetric

8. For portability reasons, the preferred method of setting the linear solver is in the FEBio configuration file, but it can also be specified in the solver section. See 9.2 for more details on configuring linear solvers.

```
<linear_solver type="pardiso"/>
```

3.3.4 Quasi-Newton Solver Parameters

Most solvers in FEBio use the quasi-Newton method for solving the nonlinear FE equations. Several different QN algorithms are implemented as described below. The specific QN method is selected using the *qn_method* property of the *solver*. The following QN methods are supported.

BFGS The Broyden-Fletcher-G-S method. (default if qn_method is not specified). This method assumes the global stiffness matrix is symmetric. It can still be used with non–symmetric matrices, but the convergence might be compromised.

Broyden The Broyden method is similar to BFGS, but works with both symmetric and non-symmetric matrices. Therefore, it is the recommended method for problems that generate a non-symmetric stiffness matrix, such as (some) contact formulations, biphasic, multi-phasic, fluid, fluid FSI.

JFNK This is an implementation of the Jacobian-Free-Newton-Krylov method. When using this option, you must also use an iterative linear solver such as FGMRES.

Both the BFGS and Broyden methods calculate the global stiffness matrix at the beginning of each time step. For each iteration, a matrix update is then done. The maximum number of such updates is set with *max_ups*. When FEBio reaches this number, or if the solution diverges and *diverge_reform* is set to 1, it reforms the global stiffness matrix (that is, it recalculates it) and factorizes it, essentially taking a "full Newton" iteration. Then FEBio continues with BFGS/Broyden iterations. The *max_refs* parameter is used to set the maximum of such reformations FEBio can do, before it fails the timestep. In that case, FEBio will either terminate or, if the auto-time stepper is enabled, retry with a smaller time step size.

The BFGS and Broyden take the following parameters.

Parameter	Description	Default
max_ups	This is the maximum number of updates between stiffness matrix reformations.	10

Note that when max_ups is set to 0, FEBio effectively will use a Full-Newton method since the stiffness matrix is reformed each iteration. In this case it is recommended to increase the number of *max_refs* (to e.g. 50), since the default value might cause FEBio to terminate prematurely when convergence is slow.

3.3.5 Solver Parameters for a Structural Mechanics Analysis

A structural mechanics analysis is defined by using the *solid* type in Module section. The solver parameters are given in the table below:

Parameter	Description	Default
dtol	convergence tolerance on displacement	0.001

When using the “CG-solid” solver, the following parameters are defined.

Parameter	Description	Default
cgmethod	Select solution algorithm (1)	0
preconditioner	Select the preconditioner (2)	0
lsiter	maximum number of line search iterations	10

Comments:

1. The solution algorithm can be:
   (a) 0 : Hager-Zhang conjugate gradient (This is the default and should be preferred since its faster.)
   (b) 1 : steepest descent
2. Set 0 for no preconditioner, 1 for diagonal stiffness preconditioner. The diagonal stiffness preconditioner greatly improves convergence when there are different size elements, mixed materials, or quadratic elements with midside nodes.

```
<solver type="CG-solid">
  <cgmethod>0</cgmethod>
  <preconditioner>1</preconditioner>
  <lsiter>10</lsiter>
</solver>
```

3.3.6 Solver Parameters for Biphasic Analysis

A biphasic analysis is defined by using the *biphasic* type in Module section. Since a biphasic analysis couples a fluid problem to a solid mechanics problem, all control parameters above can be used in a biphasic analysis. In addition, the following parameters can be defined:

Parameter	Description	Default
ptol	Specify the fluid pressure convergence tolerance	0.01

3.3.7 Solver Parameters for Solute and Multiphasic Analyses

When the type attribute of the Module section is set to *solute* or *multiphasic*, an analysis is solved that includes solute transport. All parameters for a biphasic analysis can be used (including the ones for a structural mechanics analysis). In addition, the following parameters can be specified:

Parameter	Description	Default
ctol	Specify the concentration convergence tolerance	0.01

3.3.8 Solver Parameters for Heat Analysis

A heat analysis uses the parameters defined in Section 3.3.1. However, not all parameters have an effect. In particular, any parameter related to the auto-time stepping capability of FEBio or the nonlinear solution strategy is ignored.

3.3.9 Solver Parameters for Fluid and Fluid-FSI Analyses

A fluid analysis is defined by using the *fluid* type in Module section (see Section 3.2). In addition to the common parameters, the following parameters can be specified:

Parameter	Description	Default
vtol	convergence tolerance on velocity	0.001
ftol	convergence tolerance on dilatation	0.001
rhoi	Spectral radius parameter ρ_∞	0
reform_each_time_step	Flag for reforming stiffness matrix at the start of each time step	1

A fluid-structure interaction analysis is defined by using the *fluid-FSI* type in Module section. It uses the parameters of *solid* and *fluid* analyses.

Transient fluid and fluid-FSI analyses may often run very efficiently using Broyden's method (*qnmethod* set to 1) with *max_ups* set to 50; efficiency may be further increased by setting *reform_each_time_step* to 0, which will postpone reforming the stiffness matrix at subsequent time steps until *max_ups* updates have been exhausted. When using Broyden's method with fluid and fluid-FSI analyses, set *rtol* to a non-zero value to ensure an accurate solution, for example *rtol*=0.001 (see Section 3.3.1).

The spectral radius parameter ρ_∞ determines the time integration scheme: Values in the range $0 \leq \rho_\infty \leq 1$ use the generalized $\alpha-$method (see *FEBio Theory Manual*). With $\rho_\infty = 0$, damping of higher frequency components (such as those produced by a step increase in velocity) occurs

theoretically within a single time step; in contrast, with $\rho_\infty = 1$, no damping occurs, potentially leading to instability in the solution process. When solving transient flows that exhibit eddies or shed vorticies, a value of $\rho_\infty = 0.5$ represents a good balance between too much and too little numerical damping.

3.4 Globals Section

The *Globals* section is used to define some global variables, such as global constants, solute data and solid bound molecule data, as well user-defined global variables.

3.4.1 Constants

Global constants are predefined variables that some modules may rely on and currently include the universal gas constant R [**F**·**L**/**n**·**T**], absolute temperature θ [**T**], and Faraday constant F_c [**Q**/**n**]. These constants must be expressed in units consistent with the rest of the analysis:

```
<Globals>
  <Constants>
    <R>8.314e-6</R>
    <T>298</T>
    <Fc>96485e-9</Fc>
  </Constants>
</Globals>
```

3.4.2 Solutes

In biphasic-solute, triphasic, and multiphasic analyses, a unique identifier must be associated with each solute in order to enforce consistent nodal degrees of freedom across boundaries of different materials. This unique identification is achieved by listing each solute species that appears in the entire finite element model and associating it with a unique *id*, *name*, and providing its charge number z^α, molar mass M^α, and density ρ_T^α:

```
<Globals>
  <Solutes>
    <solute id="1" name="Na">
      <charge_number>1</charge_number>
      <molar_mass>22.99</molar_mass>
      <density>0.97</density>
    </solute>
    <solute id="2" name="Cl">
      <charge_number>-1</charge_number>
      <molar_mass>35.45</molar_mass>
      <density>3.21</density>
    </solute>
    <solute id="3" name="Glc">
      <charge_number>0</charge_number>
      <molar_mass>180.16</molar_mass>
      <density>1.54</density>
    </solute>
  </Solutes>
</Globals>
```

These solute identification numbers should be referenced in the *sol* attribute of solutes when defining a biphasic-solute (Section 4.11.2), triphasic or multiphasic material (Section 4.12.2).

The molar mass and density of solutes are needed only when solutes are involved in chemical reactions. When not specified, default values for these properties are set to 1.

3.4.3 Solid-Bound Molecules

In multiphasic analyses with chemical reactions involving solid-bound molecules, a unique identifier must be associated with each such molecule in order to enforce consistent properties across the entire model. This unique identification is achieved by listing each solid-bound species that appears in the entire finite element model and associating it with a unique *id*, *name,* charge number, molar mass and density:

```
<Globals>
  <SolidBoundMolecules>
    <solid_bound id="1" name="CS">
      <charge_number>-2</charge_number>
      <molar_mass>463.37</molar_mass>
      <density>1.5<density>
    </solid_bound >
  </SolidBoundMolecules >
</Globals>
```

The *id* number should be referenced in the *sbm* attribute of solid-bound molecules when included in the definition of a multiphasic material (Section 4.12). The charge number is used in the calculation of the fixed charge density contributed by this solid-bound molecule to the overall solid matrix fixed-charge density. The density is used in the calculation of the contribution of this molecule to the referential solid volume fraction. The density and molar mass are used in the calculation of the molar volume of this molecule in chemical reactions.

3.4.4 User Variables

Users can define custom, global variables, that can be referenced in other model parameters. This offers a convenient way to parameterize a model. Global user variables can also be used in optimizations. Currently, only scalar values can be defined. User variables are defined in the *Variables* child element.

To define a global variable, specify the *var* element, add the *name* attribute to name the variable, and specify the variable's value as the tag's value.

The following example defines a global variable, called A, and assigns the value 1.23 to it.

```
<Variables>
  <var name="A">1.23</var>
</Variables>
```

As mentioned, these global variables can be used to define other model variables, for example, in variables that are defined with mathematical expressions. To use a global variable, prefix the name with 'fem.'.

```
<Material>
  <material name="mymat" type="neo-Hookean">
```

```
    <E type="math">3*fem.A</E>
    <v>0.3</v>
  </material>
</Globals>
```

3.5 Material Section

The material section is defined by the *Material* element. This section defines all the materials and material parameters that are used in the model. A material is defined by the *material* child element. This element has two attributes: *id,* which specifies a number that is used to reference the material, and *type*, which specifies the type of the material. The *material* element can also have a third optional attribute called *name*, which can be used to identify the material by a text description. A material definition might look like this:

```
<material id="1" type="isotropic elastic">
```

Or, if the optional *name* attribute is present:

```
<material id="2" type="rigid body" name="femur">
```

The material parameters that have to be entered depend on the material type. A complete list of available materials is provided in Chapter 4.

3.6 Mesh Section

The *Mesh* section contains all the mesh data, including nodal coordinates and element connectivity. It has the following sub-sections:

Nodes defines the nodal coordinates of the mesh

Elements defines the elements of the mesh

NodeSet defines a node set

Edge defines an edge, i.e. a set of line elements

Surface defines a surface, i.e. a set of facets

DiscreteSet defines a set of discrete elements (e.g. springs)

ElementSet defines an element set

SurfacePair define a surface pair that can be used by a contact definition.

At least one *Nodes* must be defined and one *Elements* section. The other sections are optional. The *NodeSet*, *Edge*, *Surface*, *DiscreteSet*, *ElementSet*, and *SurfacePair* sections define sets of nodes, edges, facets, discrete elements, elements, and surface pairs, respectively, and can be referenced by other sections of the model file. For instsance, boundary conditions can be defined by referencing the sets to which the boundary condition will be applied.

3.6.1 Nodes Section

The *Nodes* section contains nodal coordinates. It has an optional attribute called *name*. If this attribute is defined, the *Nodes* section also defines a node set.

```
<Nodes [name="<set name>"]>
```

The nodes are defined using the *node* tag which is a child of the *Nodes* section. Repeat the following XML-element for each node:

```
<node id="n">x,y,z</node>
```

The *id* attribute is the global identifier of the node and must be a unique number within the model definition. This *id* is used as a reference in the element connectivity section.

Multiple *Nodes* sections can be defined, but each node can only be defined once. For example:

```
<Nodes name="set01">
  <node id="1">0,0,0</node>
  ...
  <node id="101">1,1,1</node>
</Nodes>
<Nodes name ="set02">
  <node id="102">2,1,1</node>
  ...
  <node id="999">2,2,2</node>
</Nodes>
```

3.6.2 Elements Section

The *Elements* sections contain a list of the element connectivity data. Multiple *Elements* sections can be defined. The *Elements* section has the following attributes.

Attribute	Description
type	element type
name	unique name that identifies this domain

Each *Elements* section contains multiple *elem* elements that define the element connectivities. Each *elem* tag has a *id* attribute that defines the element number. For example, the following *Elements* section defines a list of hexahedral elements:

```
<Elements type="hex8" name="part1">
  <elem id="1">1,2,3,4,5,6,7,8</elem>
  <elem id="2">9,10,11,12,13,14,15,16</elem>
  ...
</Elements>
```

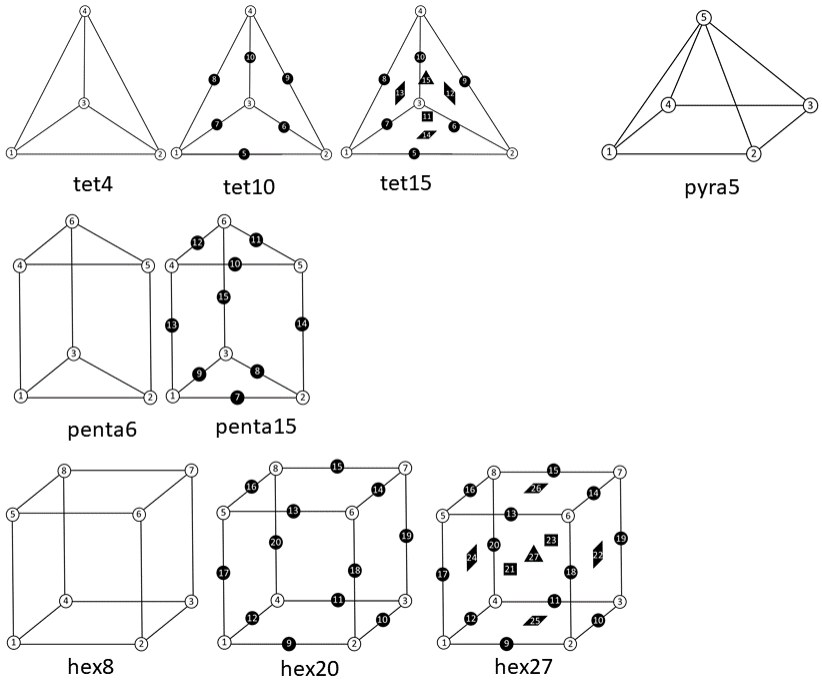
FEBio classifies elements in two categories, namely *solids* and *shells*.

3.6.2.1 Solid Elements

The following solid element types are defined:

hex8	8-node trilinear hexahedral element
hex20	20-node quadratic hexahedral elements
hex27	27-node quadratic hexahedral elements
penta6	6-node linear pentahedral (wedge) element
penta15	15-node quadratic pentahedral (wedge) element
pyra5	5-node linear pyramidal element
pyra13	13-node quadratic pyramidal element
tet4	4-node linear tetrahedral element
tet10	10-node quadratic tetrahedral element
tet15	15-node quadratic tetrahedral element

The node numbering has to be defined as in the figure below.



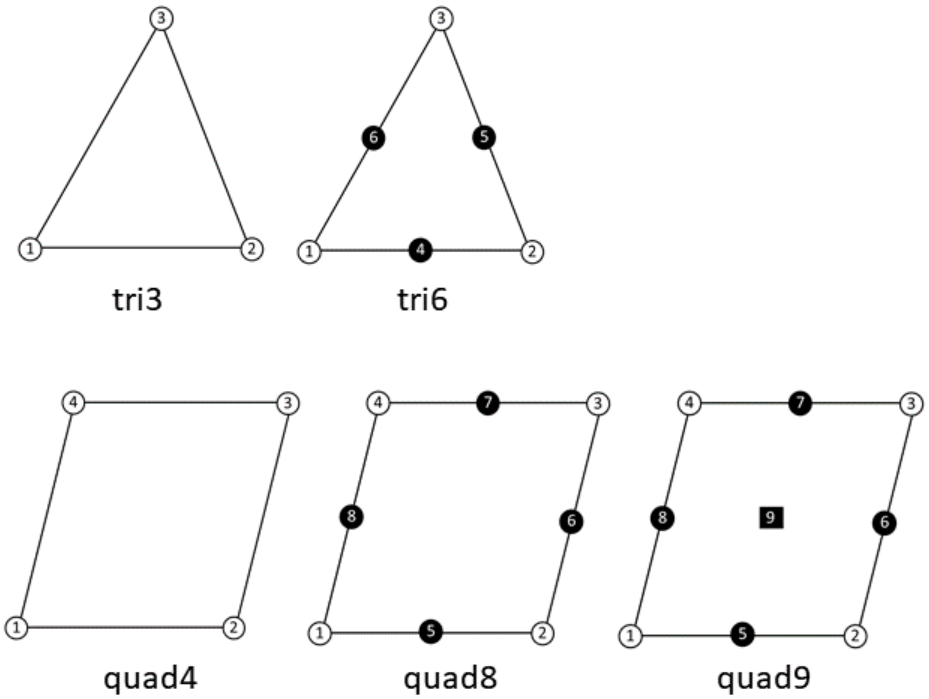
Node numbering for solid elements

3.6.2.2 Shell Elements

FEBio currently supports the following shell elements:

tri3	3-node linear triangular shell element (compatible strain)
tri6	6-node quadratic triangular shell element (compatible strain)
quad4	4-node linear quadrilateral shell element (compatible strain)
quad8	8-node quadratic quadrilateral shell element (compatible strain)
quad9	9-node quadratic quadrilateral shell element (compatible strain)
q4ans	4-node linear quadrilateral shell element (assumed natural strain)
q4eas	4-node linear quadrilateral shell element (enhanced assumed strain)

For shell elements that use a *compatible strain* formulation, the calculation of strain components is based on nodal displacements, similar to hexahedral or pentrahedral elements. Users should be aware that this compatible strain formulation is very susceptible to element locking when the shell thickness is much smaller than the shell size (e.g., when the aspect ratio is less than 0.01). Therefore, these shell formulations should be used with caution, keeping in mind this important constraint. Conversely, these shell elements perform very well when they are attached to solid elements (e.g., skin over muscle), or sandwiched between shell elements (e.g., cell membrane separating cytoplasm from extra-cellular matrix), as described in [44].



Node numbering for shell elements

The element-locking limitation of compatible strain shell formulations has motivated the development of specialized shell formulations that attempt to overcome locking. The FE literature on this subject is rather extensive and we refer the reader to the excellent review chapter by Bischoff et al. [20] on this topic. Methods for overcoming locking include the assumed natural strain (ANS) formulation for transverse shear strains [56, 16] and transverse normal strains [17, 19]. The ANS formulation may be supplemented with the enhanced assumed strain (EAS) method [74] and extended to large deformations [50, 80, 72]. FEBio includes the ANS (*q4ans*) and EAS (*q4eas*) quadrilateral shell element formulations of Vu-Quoc and Tan [80], using a seven-parameter EAS interpolation, which is otherwise substantially similar to the five-parameter interpolation presented in an earlier study by Klinkel et al. [50]. These shell elements are not suitable for attachment to a solid element, nor sandwiching between two solid elements. Since they don't experience element locking, they should be loaded more slowly than compatible strain shell elements. The most accurate of these shell formulation is *q4eas*.

By default, the nodes of a shell element define its *front* face. The location of the shell element *back* face is calculated from the nodal values of the shell thickness, along the opposite direction

of nodal normals. The nodal normal is evaluated by averaging the front face normal of all shell elements sharing that node. When a shell element is attached to a solid element (e.g., when the shell nodes also represent the nodes of one face of the solid element), FEBio automatically accounts for the shell thickness, reducing the height of the solid element in the direction of the shell normal. It is the user's responsibility to ensure that the shell thickness is less than that of the solid element; otherwise, a negative Jacobian error will be automatically generated.

Prior to FEBio 2.6, the nodes of a shell element defined its *middle* face (halfway between front and back, such that the shell element extends above and below the face defined by the shell nodes). This older formulation can be recovered by setting *shell_formulation* to 0 in the *Control* section. This setting only works with compatible strain shell formulations.

When shell domains intersect such that three or more shell elements share the same edge, as in a T-connection, users should set *shell_normal_nodal* to 0 in the *Control* section. This setting ensures that the back face of a shell element is calculated using the front face normal, instead of nodal normals, since the latter would be ill-defined for nodes belonging to such an edge.

Finally, please note that shell elements cannot be stacked in FEBio.

3.6.3 NodeSet Section

The *NodeSet* section allows users to define node sets. These node sets can then later be used in the definition of the fixed and prescribed boundary conditions. A node set is defined by the *NodeSet* tag. This tag takes one required attribute, *name*, which defines the name of the node set. A node set definition is followed by a list of node IDs. For example,

```
<NodeSet name="nodeset1">1, 2, 101, 102</NodeSet>
```

For larger nodesets, it is recommended to split the list across several lines.

```
<NodeSet name="nodeset1">
   1, 2, 3, 4,
   5, 6, 7, 8,
   9, 10, 11, 12
</NodeSet>
```

3.6.4 Edge Section

The *Edge* section allows users to define a set of edges. These edges can be used to define boundary conditions or load.s

```
<Edge name="edge1">
  <line2 lid="1">1,2</line2>
  <line2 lid="2">2,3</line2>
</Edge>
```

Here, the *lid* attribute defines the *local* id of the edge, local with respect to the edge definition. The local ids must begin at 1 and defined sequentially.

The edge elements are defined by using a tag that depends on the type of the edge element. The following edge elements are currently supported.

line2 2-node line element

line3 3-node line element

3.6.5 Surface Section

The *Surface* section allows users to define surfaces. These surfaces can then later be used to define the boundary conditions and contact definitions. A surface definition is followed by a list of surface elements, following the format described below.

```
<Surface name="named_surface">
  <quad4 id="1">1,2,3,4</quad4>
  <...>
</Surface>
```

The *Surface* takes one required attribute, namely the *name*. This attribute sets the name of the surface. This name will be used later to refer back to this surface.

The following surface elements are available:

quad4 4-node quadrilateral element

quad8 8-node serendipity quadrilater element

tri3 3-node triangular element

tri6 6-node quadratic triangular element

tri7 7-node quadratic triangular element

The value for the surface element is the nodal connectivity:

```
<quad4 id="n">n1,n2,n3,n4</quad4>
<tri3 id="n">n1,n2,n3</tri3>
```

Surface elements cannot overlap element boundaries. That is, the surface element must belong to a specific element. Surface elements do not contribute to the total number of elements in the mesh. They are also not to be confused with shell elements.

3.6.6 ElementSet Section

The *ElementSet* section can be used to define an element set. Element sets can be used to output data for only a subset of elements. An element set is defined through the *ElementSet* tag, which takes one attribute, namely *name* that specifies the name of the element set. The value of the tag is the list of element IDs. For example,

```
<ElementSet name="set01">
  1, 2, 3, 4
  5, 6, 7, 8
</ElementSet>
```

3.6.7 DiscreteSet Section

The *DiscreteSet* section defines sets of discrete elements. These sets can then be used to define e.g. springs.

```
<DiscreteSet name="springs">
  <delem>1,2</delem>
  <delem>3,4</delem>
</DiscreteSet>
```

3.6.8 SurfacePair Section

The *SurfacePair* section defines a pair of surfaces that can be used to define surface-to-surface interactions (e.g. contact).

```
<SurfacePair name="contact1">
  <primary>surface1</primary>
  <secondary>surface2</secondary>
</SurfacePair>
```

The surfaces (i.e. *surface1* and *surface2*) are surfaces defined in the *Surface* section. Because surfaces must already be defined before they can be referenced in the *SurfacePair* section, the *Surface* must be defined before the *SurfacePair* section.

3.7 MeshDomains Section

The *MeshDomains* section assigns various physical attributes to the element sets of the mesh. For each of the *Elements* sections in the *Mesh* section, define a *SolidDomain* or a *ShellDomain* section, depending on whether the elements are solid or shells. Each domain also requires a material name that defines the material properties of the domain.

3.7.1 SolidDomain Section

Use the *SolidDomain* section to define a solid domain. This section takes the following parameters:

name The name of the element set (defined via an *Elements* section). This element set must only contain solid elements.

mat The name of the material that will be assigned to this domain

type (optional) the specific formulation used for this domain.

The optional **type** attribute can be added to specify a specific solid formulation. If omitted, FEBio will determine the proper domain type by inspecting the material and element types used. However, users can explicitly set the type attribute. The following solid domain types are defined.

type	Description
elastic-solid	standard solid formulation (default for most materials) (1)
three-field-solid	A three-field formulation used for modeling (nearly) incompressible materials. (default for un
rigid-solid	Rigid solid domain. (default for rigid materials.)
udg-hex	uniform deformation gradient hex formulation
sri-solid	Formulation that uses a selective reduced integration rule
remodeling-solid	Formulation used for elastic remodeling problems.
ut4-solid	Uniform nodal strain tetrahedron. (2)

Comments:

1. For the elastic-solid domain, the elem_type attribute can be specified to further specialize the integration rule. The values of the rule depend on the specific element type. The tables below show the available integration rules for the different element types. The values marked with an asterisk (*) are the default.

- For the *hex8* element, the following values are defined.

hex8	Description
GAUSS8*	Gaussian integration using 2x2x2 integration points.
POINT6	Alternative integration rule for bricks using 6 integration point

- For the *hex20* element, the following values are defined.

hex20	Description
GAUSS8	Gaussian integration using 2x2x2 integration points.
GAUSS27*	Gaussian integration using 3x3x3 integration points.

- For the *tet4* element, the following values are allowed.

tet4	Description
GAUSS4	Gaussian integration rule using 4 integration points.
GAUSS1*	Gaussian integration rule using one integration point.

- For the *tet10* element, the following integration rules are supported.

tet10	description
GAUSS4*	Gaussian integration rule using 4 integration points
GAUSS8	Gaussian integration rule using 8 integration points
LOBATTO11	Gauss-Lobatto integration rule using 11 integration points

  The Lobatto integration rule differs from a regular Gauss integration rule in that it includes the vertices of the tetrahedral element. The *Lobatto11* integration rule uses the 10 tetrahedral nodes, plus one integration rule located at the center of the element.

- For the *tet15* element, the following integration rules are defined.

tet15	description
GAUSS8	Gaussian integration rule using 8 integration points
GAUSS11	Gaussian integration rule using 11 integration points
GAUSS15	Gaussian integration rule using 15 integration points

- For the *penta15* element, the following values are defined.

penta15	Description
GAUSS8	Gaussian integration using 2x2x2 integration points.
GAUSS21*	Gaussian integration using 3x7 integration points.

- For the *tri3* element, the following integration rules are supported.

tri3	description
GAUSS1	Gaussian integration with one integration point
GAUSS3*	Guassian integration with three integration rules.

- For the *tri6* element, the following integration rules are supported.

tri6	description
GAUSS3*	Gaussian integration with 3 integration points
GAUSS6	Gaussian integration with 6 integration points
GAUSS4	Gaussian integration with 4 integration points
GAUSS7	Gaussian integration with 7 integration points
LOBATTO7	Gauss-Lobatto integration with 7 integration points.

The GAUSS6 rule has only nonzero weights at the edge nodes, which effectively reduces this rule to a 3-node rule.

2. The ut4-solid is a special formulation for tetrahedral elements that uses a nodally averaged integration rule, as proposed by Gee et al [36]. This formulation requires additional parameters. To override the default values, use the following alternative syntax:

```
<SolidDomain name="Part1" type="ut4-solid" mat="material1">
  <alpha>0.05</alpha>
  <iso_stab>0</iso_stab>
</SolidDomain>
```

The *alpha* parameter defines the amount of "blending" between the regular tet-contribution and the nodally integrated contribution. The value must be between 0 and 1, where 0 means no contribution from the regular tet and 1 means no contribution from the nodally averaged tet. The *iso_stab* parameter is a flag that chooses between two slightly different formulations of the nodally integrated tet. When set to 0, the stabilization is applied to the entire virtual work, whereas when set to 1 the stabilization is applied only to the isochoric part. See the *FEBio Theory Manual* for a detailed description of this formulation.

3.7.2 ShellDomain Section

Use the *ShellDomain* section to define a shell domain. This section takes two parameters:

name The name of the element set (defined via an *Elements* section). This element set must only contain shell elements.

mat The name of the material that will be assigned to this domain

type (optional) the specific shell formulation

Similar to solid domains, the particular shell formulation can be requested via the **type** attribute. If omitted, FEBio will determine the formulation from the material and shell element types. The following shell domain types are available.

type	Description
elastic-shell	standard shell formulation (default for most materials) (1)
three-field-shell	A three-field formulation used for modeling (nearly) incompressible materials. (default for unc
rigid-shell	Rigid shell domain. (default for rigid materials.)
elastic-shell-old	The "old" shell formulation, used in FEBio version 2.0 and older.
elastic-shell-eas	Enhanced-Assumed-Strain shell; an alternative shell formulation for incompressible shells.
elastic-shell-ans	Assumed-Natural-Strain shell; an alternative shell formulation for incompressible shells.

The default shell thickness can be specified as a parameter of the ShellDomain section.

```
<ShellDomain name="skin">
  <shell_thickness>0.01</shell_thickness>
</ShellDomain>
```

If the shell thickness parameter is not specified, the initial shell thickness is set to zero, and it is assumed that the actual shell thickness is specified in the MeshData section.

3.8 MeshData Section

The *MeshData* section is where data is specified that can be mapped to the mesh. The data can then be applied to a parameter of the model, e.g. a material parameter, or a pressure load.

To define a mesh data section add one of the following tags.

NodeData Define data on a node set

SurfaceData Define data on a surface

ElementData Define data on an element set

The following attributes can be added.

name All mesh data sections require a name, which can be specified with the *name* attribute. This name will be used by model parameters to reference the particular mesh data.

Data maps can be defined in two ways: 1) explicitly, where the value of each node (face, element) can be specified, or 2) implicitly, via data generators that define the values of each node (face, element) procedurally.

When mesh data maps are defined explicitly, the following attributes should be specified in addition to the name attribute.

data_type defines the type of data

The following table lists the supported data types.

data type	Description
scalar	A single floating point value
vec2	A 2D vector defined as x, y
vec3	A 3D vector defined as x, y, z
mat3	A 3x3 tensor, row major sorted
mat3s	A symmetric 3x3 tensor, defined as xx, yy, zz, xy, yz, xz

If the data type is not defined, the *scalar* data type is assumed.

3.8.1 Data Generators

Data generators can be used to populate the values of a data map procedurally. To use a data generator, specify the *type* attribute. If the *type* attribute is defined, the child tags must be the parameters of the generator. For example,

```
<MeshData>
  <NodeData name="E_map" type="math">
    <math>X*X + Y*Y + Z*Z</math>
  <NodeData>
</MeshData>
```

Data generators that can be used for a specific mesh data are described in the sections below.

3.8.2 ElementData

This section defines data that can be mapped to the elements of an element set. The element set is specified via the *elem_set* attribute.

Defining maps A map is defined by using the *name* attribute to give the map a name. This name can then be used in mapped parameters (see Appendix A).

```
<MeshData>
  <ElementData name="my_map" elem_set="part1">
    <elem lid="1">1.23</elem>
    <elem lid="2">1.25</elem>
  </ElementData>
</MeshData>
```

The following data generators are available for element data sections.

surface-to-surface_map The "*surface-to-surface map*" generator defines a data field on the domain bounded by two surfaces. The data values are interpolated using a user-defined function between the surfaces.

```
<MeshData>
  <ElementData name="Emap" generator="surface-to-surface map" elem_set="Part1">
    <function type="math">x*x+1</function>
    <bottom_surface surface="surface1"/>
    <top_surface surface="surface2"/>
  <ElementData>
</MeshData>
```

special generators A few special data generators do not generate maps, but instead modify model data in a more direct way. The following table lists the currently supported special generators.

Property	Description	Data type/format
fiber	Specify a local fiber direction	vec3/const
mat_axis	Specify local element material axes	vec3/const
shell thickness	Specify the shell element thickness	float/shape

For example, to assign shell thicknesses to an element set, use the following syntax.

```
<MeshData>
  <ElementData type="shell thickness" elem_set="part1">
    <elem lid="1">1.0, 1.0, 1.0, 1.0</elem>
    <elem lid="2">1.0, 1.0, 1.0, 1.0</elem>
  </ElementData>
</MeshData>
```

3.8.3 SurfaceData

This section allows users to define data that can be mapped to the surfaces of certain boundary conditions and loads. This section only defines user-defined data maps, i.e. the *name*, *data_type* and *data_format* attributes must be used. The surface is defined via the *surface* attribute.

```
<SurfaceData name="values" surface="surface1">
  <face lid="1">1.23</face>
  <face lid="2">3.45</face>
</SurfaceData>
```

The surface definition is defined in the *Mesh* section. Note that the ids refer to the local ids of the surface facets. By default, the *scalar* data type and *const* data format are assumed so only one value per facet is expected. Other data types and formats can be specified with the *data_type* and *data_format* attributes.

```
<SurfaceData name="data" data_type="vec3" surface="surface1">
  <face lid="1">0.0, 1.0, 0.0</face>
  <face lid="2">0.0, 2.0, 0.0</face>
</SurfaceData>
```

Optionally, the surface data may be described with a data generator. The data generator is specified via the *type* attribute. The following data generators are available for surface data.

parabolic_map The "*parabolic map*" defines a scalar data field with a parabolic profile on a surface. The data is determined by solving a heat-type problem with a constant source and homogeneous boundary conditions on the edge of the surface.

```
<MeshData>
  <SurfaceData name="pressure" type="parabolic map" surface="surface1">
    <value>1.0</value>
  <SurfaceData>
</MeshData>
```

3.8.4 EdgeData

This section allows users to define user-defined data maps that will be mapped to an edge defined in the Mesh section.

```
<EdgeData name="data" edge="edge1">
  <edge lid="1">1.0</edge>
  <edge lid="2">2.0</edge>
</EdgeData>
```

Notice that the *local* IDs are required here.

3.8.5 NodeData

This section allows users to define data that will be mapped to node sets.

```
<NodeData name="data" node_set="set1">
  <node lid="1">1.0</node>
  <node lid="2">2.0</node>
</NodeData>
```

For node data the *data_format* attribute, if specified, will be ignored.

3.9 Mesh Adaptor Section

The FEBio Adaptive Mesh Refinement framework (FEAMR, pronounced “femur”) allows users to make modifications to the mesh during the analysis of the model. There are currently two mesh adaptor applications supported, namely mesh erosion, which removes elements from the mesh, and remeshing, which can be used to adaptively refine or coarsen the mesh based on the solution. The mesh adaptors are controlled via user-defined criteria that determine the conditions under which to modify the mesh.

A mesh adaptor is applied at the end of a timestep, after the Newton-iterations and augmentations have converged. If the mesh adaptor modifies the mesh, then the timestep is solved again, using the new mesh. Several adaptors define the *criterion* property, which evaluates the variable or metric that is used by the adaptor, and optionally, can be used to filter the elements to which the adaptor will be applier. How the criterion is used, depends on the adaptor. For instance, the erosion adaptor uses it to identify the elements to deactivate. Remeshing adaptors will use it to set the new element size. It is allowed to use several mesh adaptors in a single model, but keep in mind that the order in which the mesh adaptors are listed in the input file is important. For instance, if two mesh adaptors are defined, the second adaptor will operate on the mesh created by the first adaptor. Mesh adaptors used for adaptive refinement will also map data between the old mesh and the new mesh. User data, defined in the MeshData section, is always mapped between meshes. If the model is history-dependent, it might be important to map additional data. For instance, a visco-elastic material needs to map stress-history variables. By default, mapping data (except for user-data) is not done unless the user sets a specific flag (map_data). New mesh adaptors and adaptor criteria can be defined via the plugin framework.

3.9.1 The MeshAdaptor Section

Mesh adaptors are added to the FEBio Input file (version 3.0 and above) in a new section, called **MeshAdaptor**. This section must appear after the Mesh and MeshDomains sections. The MeshAdaptor section may also be defined in a *step* section in order to apply the adaptor only during that step. Inside the MeshAdaptor section, each mesh adaptor is defined via the **mesh_adaptor** tag. This tag allows the following attributes:

- **type**: set the type of the mesh adaptor (1). [Required]
- **elem_set**: The element set to apply the adaptor to (2). [Optional]

Comments:

1. The **type** attribute defines the mesh adaptor. It must be one of the mesh adaptors defined in Table 1, below.

2. The **elem_set** attribute is optional, but can be used to apply the adaptor to only a part of the mesh. This is useful to limit the effect of an adaptor to a part of the mesh. The element set must reference a domain (i.e. defined via Elements) of a mesh, or a set of domains. It cannot be an element set defined through the ElementSet tag. (Note that not all mesh adaptors support this attribute.) If omitted, the adaptor is applied to the entire mesh.

The sections below list the properties of the available mesh adaptors and the adaptor criteria.

type	Description
erosion	The erosion adaptor removes elements from a mesh.
mmg_remesh	Adaptive mesh refinement of tetrahedral mesh using the MMG library.
hex_refine2d	Refine a 2D hexahedral mesh.
tet_refine	Refine a linear tetrahedral mesh (Whole mesh refinement only)
hex_refine	Refine a hexahedral mesh.

3.9.1.1 The erosion adaptor

The erosion adaptor can be used to remove or deactivate elements in a mesh, based on a user-defined criterion. It defines the following properties:

- **max_iters**: Set the max number of adaption iterations, for each time step.
- **remove_islands**: If true, removes small element sets that have become disconnected from the main mesh.
- **max_elems**: Set the max number of elements to erode, per adaptation.
- **sort**: sort the element values returned by the criterion.
- **criterion**: Set the criterion that determines the element metric to evaluate and which elements to erode.

Example:
In this example, elements that have an (integration point averaged) effective stress larger than 0.3, are eroded. Only three elements are eroded per adaptation cycle.

```
<mesh_adaptor type="erosion">
  <max_iters>1</max_iters>
  <max_elems>3</max_elems>
  <sort>1</sort>
  <criterion type="min-max filter">
    <min>0.3</min>
    <clamp>0</clamp>
    <data type="stress"/>
  </criterion>
</mesh_adaptor>
```

3.9.1.2 The mmg_remesh adaptor

The mmg_remesh adaptor can be used for adaptive mesh refinement of linear tetrahedral meshes.

- **max_iters**: The maximum number of adaption iterations per time step.
- **max_elements**: The maximum number of elements. If the number of elements of the mesh is larger, the adaptor is not applied.
- **min_element_size**: Set the desired minimal element size.

- **hausdorff**: The Hausdorff distance. This is used to preserve curved sections. (default = 0.01).

- **gradation**: This parameter sets the spatial gradient of element size. (default = 1.3). Value should be larger than 1.

- **mesh_coarsen**: Allow mesh-coarsening

- **normalize_data**: normalize the element values returned by the criterion between 0 and 1.

- **relative_size**: Sets whether the size function is to be interpreted as a relative size function (i.e. a scale function to the current element size), or an absolute element size metric.

- **criterion**: The metric used to assign values to elements. (See section 4)

- **size_function**: set the mesh size function. This function is applied to the (optionally normalized) element values returned by the criterion.

Example:

```
<mesh_adaptor type="mmg_remesh">
  <max_iters>1</max_iters>
  <criterion type="stress"/>
  <relative_size>1</relative_size>
  <normalize_data>0</normalize_data>
  <mesh_coarsen>0</mesh_coarsen>
  <size_function type="step">
    <x0>10</x0>
    <left_val>1</left_val>
    <right_val>0.5</right_val>
  </size_function>
</mesh_adaptor>
```

3.9.2 Mesh Adaptor Criteria

The purpose of the mesh adaptor criteria is to assign values to elements. Some adaptors also act as filters, that limit the effect of the adaptor to certain elements. These values will then be used by the mesh adaptor to modify the mesh. How these values are used depends on the adaptor. For instance, the erosion uses it to decide which elements to erode. The mesh refinement adaptors use it to define the size function.

The following criteria are currently available. A detailed description of these adaptor criteria is given below.

type	Description
max_variable	Evaluates average of element's nodal values of a model primary variable.
element_selection	Sets the element selection to which the adaptor should be applied (only for erosion)
stress	Evaluate the (integration point average) effective stress of each element.
damage	Evaluate the (integration point average) damage value.
relative error	Evaluate a relative error measure.
spring force	Evaluate the axial force of the spring.
spring stretch	Evaluate the stretch of a spring.
math	Specify the element value based on a mathematical expression.
min-max filter	Selects elements whose value falls within a min-max range.

3.9.2.1 max_variable

The max_variable criterion evaluates elements based on a primary variable's degrees of freedom.

- **dof**: The degree of freedom that defines the variable to inspect.

3.9.2.2 element_selection

Sets the selected elements directly.

- **element_list:** a comma-separated list of element indices.

3.9.2.3 math

Sets the scale factor for each element.

- **scale**: the scale factor to use. This can be a constant, or a mathematical expression of position (use X, Y, Z for the nodal coordinates), or a scalar map defined in the MeshData section.

Example:

```
<mesh_adaptor type="mmg_remesh">
  <max_iters>1</max_iters>
  <criterion type="math">
    <math>1*(1 - (X^2+Y^2)) + (X^2+Y^2)*(0.5*(Z-1)^2 - Z*(Z - 2))</math>
  </criterion>
</mesh_adaptor>
```

3.9.2.4 stress

This criterion selects elements that exceed a maximum stress value.

- **metric**: The stress measure to use: 0=effective stress, 1=max principal stress.

3.9.2.5 max_damage

This criterion evaluates the maximum damage value for each element. This can only be used with materials that define the damage property.

3.9.2.6 relative error

This criterion evaluates a relative error metric for each element.

- **error:** If nonzero, this criterion will evaluate a relative size metric for each element. Otherwise, the actual relative error is evaluated.
- **data:** Specify the metric to evaluate the element values. Any of the other criteria defined above can be used.

3.9.2.7 min-max filter

This criterion evaluates a metric for each element, but then filters out the element that have a value within a min-max range.

- **min**: set the minimum value of the filter
- **max**: set the maximum value of the filter
- **clamp**: If true, the element values will be clamped to the range.
- **data**: Specify the metric to evaluate the element values. Any of the other criteria defined above can be used.

3.9.3 Limitations

Please take into account the following limitations of the FEAMR framework.

1. Although contact can be used with the FEAMR framework, currently no contact data is mapped from the old mesh to the new mesh. It is advised to only use penalty-based contact interfaces.
2. It is not advised to erode elements that are in contact, since this will not produce the correct result.
3. Mapping history-data is supported (if the proper flag is set), but can be very time consuming since FEBio maps all material point data between meshes.
4. The FEAMR framework does not work with the auto time-stepper. It is advised to use fixed time stepping when using mesh adaptation.

3.10 Initial Section

The *Initial* section defines all initial conditions that may be applied to the analysis. An initial condition is defined via the *ic* element and requires the *type* attribute and *node_set* attribute. The *type* attribute defines the particular initial condition that will be applied. The *node_set* attribute defines the set of nodes that this initial condition affects.

type	Description
init_dof	Set the initial value of a specific degree of freedom. (obsolete)
velocity	Set the initial value for the nodal velocities.
shell velocity	Set the initial value for the shell (back) nodal velocities.
initial fluid velocity	Set the initial values for the fluid velocity (in fluid analysis)
prestrain	Used to initialize the prestrain analysis.

Example:

```
<ic type="init_dof" node_set="set1">
  <dof>x</dof>
  <value>1.0</value>
</ic>
```

3.10.1 The Prestrain Initial Condition

The prestrain initial condition can be used to accomplish one of two things. It can be used to convert a forward analysis in an equivalent prestrain analysis. Or it can be used to fix the prestrain gradient. In either case the current geometry is used as the new reference geometry of the following analysis step.

parameter	description	initial value
init	Initialize prestrain field	1 (=true)
reset	Make current geometry the reference geometry of the following steps	1 (=true)

The following example shows how a forward analysis can be converted to a prestrain analysis.

```
<ic type="prestrain">
  <init>1</init>
  <reset>1</reset>
</ic>
```

Note that this in itself may not be sufficient to define the equivalent prestrain analysis. For instance, if the forward model applied prescribed displacements, then these boundary conditions must be replaced with fixed boundary conditions.

If the *init* parameter is set to 0 (false), then the prestrain gradient will be fixated in the current geometry. This means that any remaining distortion will be applied to the prestrain correction factor before the current geometry is converted to the new reference geometry.

3.10.2 Fluid Pressure Initial Condition

The fluid pressure initial condition can be used in biphasic and multiphasic analyses, and in fluid, fluid-FSI, and fluid-solutes analyses. In all cases it sets the initial value of the effective fluid pressure $\tilde{p}$. If the analysis includes solutes (multiphasic or fluid-solutes), it is the user's responsibility to evaluate $\tilde{p}$ using eq.(4.12.7), knowing the values of the universal gas constant R and absolute temperature T specified in the Globals section (Section 3.4), and the values of initial effective solute concentrations $\tilde{c}$ in the given multiphasic or fluid-solutes domain.

In biphasic and multiphasic analyses the value of $\tilde{p}$ is prescribed directly as a nodal degree of freedom. In a fluid, fluid-FSI, or fluid-solutes analysis, the effective fluid pressure $\tilde{p}$ is converted internally to the corresponding initial value of the fluid dilatation e^f, based on the constitutive models presented in eq.(4.16.2).

3.11 Boundary Section

The *Boundary* section defines all fixed and prescribed boundary conditions that may be applied to the geometry. Individual boundary conditions are defined via the *bc* tag and the particular boundary condition is defined via the *type* attribute. The following boundary condition types are currently supported.

type	description	mo
`zero displacement`	Fix the x, y, or z displacement degree of freedom	sc
`zero shell displacement`	Fix the x, y, or z shell (back) displacement dof	sc
`zero fluid pressure`	Fix the biphasic or multiphasic effective fluid pressure $\tilde{p}$	bipl
`zero concentration`	Fix the effective solute concentration $\tilde{c}$ (multiphasic or fluid-solutes)	so
`zero fluid dilatation`	Fix the fluid dilatation e^f	fl
`prescribed displacement`	Prescribe the x, y, or z displacement dof	sc
`prescribed shell displacement`	Prescribe the x, y, or z shell (back) displacement dof	sc
`prescribed deformation`	Prescribe the motion via a deformation gradient	sc
`prescribed fluid pressure`	Prescribe the biphasic or multiphasic effective fluid pressure $\tilde{p}$	bipl
`prescribed concentration`	Prescribe the effective solute concentration $\tilde{c}$	so
`prescribed fluid dilatation`	Prescribe the fluid dilatation e^f	fl
`normal displacement`	Prescribe the motion normal to a surface (in reference configuration)	sc
`rigid deformation`	Prescribe the motion via a rigid motion	sc
`rigid`	The nodes are attached to a rigid body	sc
`fluid pressure`	Prescribe the actual fluid pressure p on a surface	mu
`fluid resistance`	Prescribe a fluid resistance R such that $p = Rq$, $q =$volumetric flux	mu
`linear constraint`	The motion is constrained to satisfy a linear constraint	(a

Most of these boundary conditions require the node set to which to apply the boundary condition to. That nodeset can be defined via the *node_set* attribute. The value of this attribute should be set to the name of a nodeset defined in the Mesh section. See Section 3.6.3 for more information on how to define node sets. Nodesets can also be defined via surfaces or element sets. To define a node set via a surface, use the following syntax.

```
node_set="@surface:surface1"
```

Here, *surface1* is a surface defined in the *Mesh* section. Similarly, defining a node set via an element set is done using *@elem_set:.*

3.11.1 Zero Displacement

To fix the nodal displacement degrees of freedom (dof) use the *zero displacement* boundary condition. This element has two required attributes: the *type* is set to “zero displacement” and the node set to which this boundary condition applies is defined vua *node_set*. The optional *name* attribute can be used to give the boundary condition a name.

This boundary condition has three parameters, one for each degree of freedom. Set the value of each parameter to 1 to constrain the corresponding dof.

```
<bc type="zero displacement" node_set="nodeset1">
  <x_dof>1</x_dof>
  <y_dof>1</y_dof>
  <z_dof>1</z_dof>
</bc>
```

3.11.2 Zero Shell Displacement

To fix the nodal back-displacement degrees of freedom (dof) of a shell use the *zero shell displacement* boundary condition. This element has two required attributes: the *type* is set to "zero shell displacement" and the node set to which this boundary condition applies is defined vua *node_set*. The optional *name* attribute can be used to give the boundary condition a name.

This boundary condition has three parameters, one for each degree of freedom. Set the value of each parameter to 1 to constrain the corresponding dof.

```
<bc type="zero shell displacement" node_set="nodeset1">
  <sx_dof>1</sx_dof>
  <sy_dof>1</sy_dof>
  <sz_dof>1</sz_dof>
</bc>
```

This boundary condition can only be applied to nodes that belong to shells.

3.11.3 Zero Fluid Pressure

To fix the effective fluid pressure $\tilde{p}$ in a biphasic or multiphasic analysis, use the *zero fluid pressure* boundary condition.

```
<bc node_set="set01" type="zero fluid pressure"/>
```

3.11.4 Zero Concentration

To fix an effective solute concentration $\tilde{c}$ degree of freedom (c_dof) in a multiphasic or fluid-solutes analysis, use the *zero concentration* boundary conditions.

```
<bc node_set="set01" type="zero concentration">
  <c_dof>c1</c_dof>
</bc>
```

3.11.5 Zero Fluid Dilatation

To fix the fluid dilatation e^f in a fluid or fluid-FSI or fluid-solutes analysis, use the *zero fluid dilatation* boundary condition.

```
<bc node_set="set01" type="zero fluid dilatation"/>
```

Zero fluid dilatation is the same as setting the effective fluid pressure $\tilde{p}$ to zero in fluid analyses. If you want to set the actual fluid pressure p to some value (such as zero) in a fluid analysis, use the *fluid pressure* boundary condition (Section 3.11.16).

3.11.6 Prescribed Displacement

The prescribed displacement boundary condition can be used to prescribe the value of the nodal displacement degrees of freedom. This boundary condition takes the following parameters.

parameter	description	default
`dof`	The displacement dof to prescribe, x, y, or z	-
`value`	The prescribed value	0
`relative`	Flag that defines whether the value is absolute or relative (1)	0 (off)

Comments:

1. If the relative flag is set to 1, then the value is relative with respect to the current position of the node at the time the boundary condition becomes active. This only has an effect in multi-step analyses.

```
<bc type="prescribed displacement" node_set="set1">
  <dof>x</dof>
  <value lc="1">1.0</value>
  <relative>0</relative>
<bc>
```

Although the *prescribe displacement* element with a value of zero can also be used to fix a certain nodal degree of freedom, the user should use the *zero displacement* boundary condition whenever possible, since this option causes the equation corresponding to the constrained degree of freedom to be removed from the linear system of equations. This results in fewer equations that need to be solved for and thus reduces the run time of the FE analysis.

3.11.7 Prescribed Shell Displacement

The *prescribed shell displacement* boundary condition can be used to prescribe the value of the nodal back-displacement degrees of freedom of a shell. This boundary condition takes the following parameters.

parameter	description	default
`dof`	The displacement dof to prescribe, sx, sy, or sz	-
`value`	The prescribed value	0
`relative`	Flag that defines whether the value is absolute or relative (1)	0 (off)

Comments:

1. If the relative flag is set to 1, then the value is relative with respect to the current position of the node at the time the boundary condition becomes active. This only has an effect in multi-step analyses.

```
<bc type="prescribed shell displacement" node_set="set1">
  <dof>sx</dof>
  <value lc="1">1.0</value>
  <relative>0</relative>
<bc>
```

Although the *prescribe shell displacement* element with a value of zero can also be used to fix a certain nodal degree of freedom, the user should use the *zero shell displacement* boundary condition whenever possible, since this option causes the equation corresponding to the constrained degree of freedom to be removed from the linear system of equations. This results in fewer equations that need to be solved for and thus reduces the run time of the FE analysis.

3.11.8 Prescribed Fluid Pressure

The *prescribed fluid pressure* boundary condition can be used to prescribe the effective fluid pressure $\tilde{p}$ on the boundary of a biphasic or multiphasic material. This boundary condition takes the following parameters.

parameter	description	default
value	The prescribed value	0
relative	Flag that defines whether the value is absolute or relative (1)	0 (off)

Comments:

1. If the relative flag is set to 1, then the value is relative with respect to the current position of the node at the time the boundary condition becomes active. This only has an effect in multi-step analyses.

```
<bc type="prescribed fluid pressure" node_set="set1">
  <value lc="1">1.0</value>
  <relative>0</relative>
<bc>
```

Although the *prescribed fluid pressure* element with a value of zero can also be used to fix the effective fluid pressure, the user should use the *zero fluid pressure* boundary condition whenever possible, since this option causes the equation corresponding to that degree of freedom to be removed from the linear system of equations. This results in fewer equations that need to be solved for and thus reduces the run time of the FE analysis.

3.11.9 Prescribed Concentration

The *prescribed concentration* boundary condition can be used to prescribe the value of the effective solute concentration $\tilde{c}$ in a multiphasic or fluid-solutes analysis. This boundary condition takes the following parameters.

parameter	description	default
dof	The solute concentration dof to prescribe, c1, c2, etc.	-
value	The prescribed value	0
relative	Flag that defines whether the value is absolute or relative (1)	0 (off)

Comments:

1. If the relative flag is set to 1, then the value is relative with respect to the current effective solute concentration at the time the boundary condition becomes active. This only has an effect in multi-step analyses.

```
<bc type="prescribed concentration" node_set="set1">
  <dof>c1</dof>
  <value lc="1">1.0</value>
  <relative>0</relative>
<bc>
```

Although the *prescribed concentration* element with a value of zero can also be used to fix that degree of freedom, the user should use the *zero concentration* boundary condition whenever possible, since this option causes the equation corresponding to the constrained degree of freedom to be removed from the linear system of equations. This results in fewer equations that need to be solved for and thus reduces the run time of the FE analysis.

3.11.10 Prescribed Deformation

The *prescribed deformation* boundary condition can be used to prescribe the displacement of a node using the following equation.

$$\boldsymbol{u} = s\left(\boldsymbol{F} - \boldsymbol{I}\right)\boldsymbol{X}$$

Here, $\boldsymbol{u}$ is the displacement vector, $\boldsymbol{X}$ the reference position vector, $\boldsymbol{F}$a "deformation gradient", $\boldsymbol{I}$ the identity tensor, and s a scale factor.

The following parameters are defined.

parameter	description	default
scale	The scale factor s	1
F	The deformation gradient	$\boldsymbol{I}$

```
<bc type="prescribed deformation" node_set="set1">
  <scale lc="1">0.1</scale>
  <F>2,0,0, 0,2,0, 0,0.25,0</F>
<bc>
```

3.11.11 Prescribed Fluid Dilatation

The *prescribed fluid dilatation* boundary condition can be used to prescribe the fluid dilatation e^f on the boundary of a fluid, fluid-FSI or fluid-solutes material. This boundary condition takes the following parameters.

parameter	description	default
value	The prescribed value	0
relative	Flag that defines whether the value is absolute or relative (1)	0 (off)

Comments:

1. If the relative flag is set to 1, then the value is relative with respect to the current position of the node at the time the boundary condition becomes active. This only has an effect in multi-step analyses.

```
<bc type="prescribed fluid dilatation" node_set="set1">
  <value lc="1">-1e-8</value>
  <relative>0</relative>
<bc>
```

Although the *prescribed fluid dilatation* element with a value of zero can also be used to fix the effective fluid pressure, the user should use the *zero fluid dilatation* boundary condition whenever possible, since this option causes the equation corresponding to that degree of freedom to be removed from the linear system of equations. This results in fewer equations that need to be solved for and thus reduces the run time of the FE analysis.

3.11.12 Normal Displacement

The *normal displacement* boundary condition prescribes the displacement of a node along the normal vector to the surface on which the node lies.

$$\boldsymbol{u} = s\boldsymbol{N}$$

Here, $\boldsymbol{u}$ is the displacement vector, $\boldsymbol{N}$ is the normal to the surface in the reference configuration, and s is a scale factor.

Note that this boundary condition requires a *surface* attribute, instead of the *node_set* attribute. The following parameters are defined.

parameter	description	default
scale	The scale factor s	0
surface_hint	Flag indicating the type of surface (1)	0

Comments:

1. The *surface_hint* parameter indicates the shape of the surface. Set this value to 1 if the surface is spherical. This will calculate a more accurate normal vector.

```
<bc type="normal displacement" surface="surf1">
  <scale lc="1">0.1</scale>
  <surface_hint>1</surface_hint>
<bc>
```

3.11.13 Rigid Deformation

The *rigid deformation* boundary condition can be used to prescribe the motion of a nodeset via a rigid transformation. The displacement is given by the following equation.

$$\boldsymbol{u} = \boldsymbol{Q}\left(\boldsymbol{X} - \boldsymbol{r}_0\right) + \boldsymbol{r}_t - \boldsymbol{X}$$

Here, $\boldsymbol{u}$ is the displacement vector, $\boldsymbol{Q}$ is a rotation tensor, $\boldsymbol{X}$ is the initial position of the node, and $\boldsymbol{r}_0$ and $\boldsymbol{r}_t$ are the reference and current position respectively of the rigid coordinate system. The following parameters are defined.

parameter	description	default
pos	Position vector of the rigid coordinate system	{0,0,0}
rot	Rotation vector of the rigid coordinate system (1)	{0,0,0}

Comments:

1. The rotation of the rigid coordinate system is given by a rotation vector. The direction of this vector defines the (instantaneous) axis or rotation, and the magnitude is the rotation angle (in radians) around this axis.

3.11.14 Rigid

A node set can be attached to a rigid body using the *rigid* boundary condition. Rigid nodes are not assigned degrees of freedom.

```
<bc type="rigid" node_set="set1">
  <rb>2</rb>
</bc>
```

The *rb* parameter defines the material (which must be a “rigid body” material) that in turn defines the rigid body. The node set must be defined in the *Mesh* section.

3.11.15 Multiphasic Fluid Pressure

In a multiphasic mixture the nodal degree of freedom for fluid pressure represents the effective fluid pressure $\tilde{p}$, which is related to the actual fluid pressure p according to eq.(4.12.7). One may assign the value of p directly on a boundary, by using the *actual fluid pressure* boundary condition.

```
<bc surface="FluidPressure1" type="actual fluid pressure">
  <pressure>0</pressure>
  <shell_bottom>0</shell_bottom>
</bc>
```

This surface load evaluates p from nodal values of $\tilde{p}$ on the surface, and using the average values of Φ and c^α at the integration points of the underlying element. This surface load is particularly useful when the effective solute concentrations $\tilde{c}^\alpha$ are not prescribed as boundary conditions on that surface, but evolve with the analysis solution. the *shell_bottom* boolean flag (0=false, 1=true) should be set to true when prescribing this pressure on the bottom of a multiphasic shell domain.

3.11.16 Fluid Pressure

To set the actual fluid pressure p in a fluid, fluid-FSI, or fluid-solutes analysis, use the *fluid pressure* boundary condition.

```
<bc type="fluid pressure" surface="FluidPressure1">
  <pressure lc="1">100</pressure>
</bc>
```

The optional *lc* parameter associates a load curve (a user-defined scale factor that evolves with time) which multiplies the value of <pressure>. The surface must be defined in the *Mesh* section. For a biphasic analysis the actual fluid pressure is equal to the effective fluid pressure ($p = \tilde{p}$), in which case use *prescribed fluid pressure* (Section 3.11.8).

3.11.17 Fluid Resistance

In a fluid, fluid-FSI or fluid-solutes analysis, this boundary condition prescribes an elastic pressure $p = R\,q + p_d$ on a surface, where R is the flow resistance and q is the volumetric flow rate across the surface (calculated internally); p_d is an optional offset pressure. The pressure p is converted to a dilatation and prescribed as an essential boundary condition at the nodes of the facets.

```
<bc type="fluid resistance" surface="surface1">
  <R lc="1">7.93e+07</R>
  <pressure_offset lc="2">10640</pressure_offset>
<bc>
```

3.11.18 Fluid RCR

This surface load models a three-element Windkessel outflow condition on a surface in a fluid, fluid-FSI or fluid-solutes analysis, as described in [79]. It prescribes an elastic pressure p on a surface, which satisfies the ordinary differential equation

$$p + \tau\frac{dp}{dt} = (R_d + R)\,q + R\tau\frac{dq}{dt} + p_d + \tau\frac{dp_d}{dt}, \quad \tau = R_d C$$

where R is the upstream flow resistance (series resistance), R_d is the downstream flow resistance and C is the downstream flow capacitance (parallel R_dC element downstream of R), whereas q is the volumetric flow rate across the surface (calculated internally); p_d is an optional downstream offset pressure, which may be associated with a loadcurve to produce a function of time. The actual pressure p obtained by solving this differential equation is converted to a dilatation and prescribed as an essential boundary condition at the nodes of the facets. Since the pressure is the solution of an ordinary differential equation, the user may also optionally specify the initial condition for the pressure.

```
<bc name="FluidRCR1" type="fluid RCR" surface="FluidRCR1">
  <R>1</R>
  <Rd>10</Rd>
  <capacitance>0.5</capacitance>
  <pressure_offset>0</pressure_offset>
  <initial_pressure>0</initial_pressure>
</bc>
```

By setting $C = 0$ we recover the fluid resistance surface load described in Section 3.11.17, with flow resistance equal to $R_d + R$. By setting R to zero we can have simulate a parallel R_dC circuit.

Internally, the ordinary differential equation presented above is solved numerically using a standard finite difference scheme,

$$p_{n+1} = \left(\frac{R_d}{1 + \frac{\tau}{\Delta t}} + R \right) q_{n+1} + \frac{\tau}{\Delta t + \tau} \left(p_n - p_{d,n} - R q_n \right) + p_{d,n+1}$$

where f_{n+1} is the value of any function $f\,(t)$ at the current time step t_{n+1}, f_n is its value at the previous time step t_n, and $\Delta t = t_{n+1} - t_n$ is the time increment.

3.11.19 Linear Constraints

A linear constraint can be used to couple nodal degrees of freedom. The linear constraint has one dependent degree of freedom u^*, and several independent degrees of freedom u_i. The linear constraint is defined as follows:

$$u^* = c_0 + c_1 u_1 + \ldots + c_n u_n$$

Here, the c_i are user-specified scale factors and $c_0$can be used to specified an offset.

The dependent degree of freedom u^*will be removed from the linear system of equations. Consequently, this nodal degree of freedom should not be used in any other boundary condition.

To define a linear constraint, set the *type* attribute to *linear constraint*. The dependent degree of freedom, i.e. the dof that will be removed, is specified via a node number and a dof identifier. Each child dof is defined via the *child_dof* tag and similarly requires the corresponding node and dof identifier, as well as the value parameter.

```
<bc type="linear constraint">
  <node>5</node>
  <dof>x</dof>
  <child_dof>
    <node>8</node>
```

```
      <dof>x</dof>
      <value>1.0</value>
    </child_dof>
    <child_dof>
      <node>9</node>
      <dof>x</dof>
      <value>-1.0</value>
    </child_dof>
  </bc>
```

3.12 Rigid Section

The *Rigid* section is used to define all rigid constraints, rigid body initial kinematics, and rigid connectors. The motion of a point $\mathbf{x}$ connected to a rigid body is given by

$$\mathbf{x} = \mathbf{r}(t) + \mathbf{R}(t) \cdot \mathbf{Z} = \mathbf{r}(t) + \mathbf{z}(t) \tag{3.12.1}$$

where $\mathbf{r}(t)$ is the position of the rigid body center of mass and $\mathbf{R}(t)$ is the body's (proper orthogonal) rotation tensor, which satisfies $\mathbf{R}(t_0) = \mathbf{I}$ at the initial time t_0; here, $\mathbf{z}(t) = \mathbf{R}(t) \cdot \mathbf{Z}$ is the position vector of the point relative to the body's center of mass, and $\mathbf{X} = \mathbf{r}(t_0) + \mathbf{Z}$ is the initial position. In FEBio, the rotation is evaluated as $\mathbf{R}(t) = \exp[\boldsymbol{\xi}(t)]$, where $\boldsymbol{\xi}(t)$ is the material rotation of the rigid body from its reference configuration. In practice, $\boldsymbol{\xi}$ is stored as a quaternion to facilitate the multiplication of rotation tensors. The exponential map $\exp[\boldsymbol{\xi}(t)]$ is given in full form as

$$\mathbf{R}(t) = \exp[\boldsymbol{\xi}(t)] = \cos\xi\,\mathbf{I} - \sin\xi\,\boldsymbol{\mathcal{E}} \cdot \mathbf{n} + (1 - \cos\xi)\,\mathbf{n} \otimes \mathbf{n} \tag{3.12.2}$$

where $\boldsymbol{\mathcal{E}}$ is the third-order permutation pseudo-tensor with rectangular components ε_{ijk} (permutation symbol). Here, $\mathbf{n}$ is the unit vector along the axis of rotation and ξ is the angle of rotation about $\mathbf{n}$, such that $\boldsymbol{\xi} = \xi\mathbf{n}$. Both ξ and $\mathbf{n}$ may evolve over time. Either none or all the components of $\boldsymbol{\xi}(t)$ must be prescribed (or fixed) for a rigid body.

The Rigid section can contain the following subsections.

rigid_bc Define a rigid constraint.

rigid_ic Define a rigid initial condition.

rigid_load Define a load on a rigid body.

rigid_connector Define a connection or joint between two rigid bodies.

3.12.1 Rigid Constraints

The rigid constraints section allows users to prescribe the kinematics applied to a single rigid body. The particular constraint is defined via the *type* attribute.

The following rigid constraints can be defined.

type	Description
rigid_fixed	Fix one or more rigid degrees of freedom.
rigid_prescribed	Prescribed the value of a rigid degree of freedom

3.12.1.1 Rigid_fixed Constraint

The *rigid_fixed* constraint fixes one or multiple rigid degrees of freedom. It requires the following parameters.

parameters	Description
rb	The rigid material name (1)
Rx_dof	fix x-displacement degree of freedom
Ry_dof	fix y-displacement degree of freedom
Rz_dof	fix z-displacement degree of freedom
Ru_dof	fix x-rotational degree of freedom
Rv_dof	fix y-rotational degree of freedom
Rw_dof	fix z-rotational degree of freedom

Comments:

1. The name of the rigid material defined in the Material section.

The following example fixes the displacement degrees of freedom.

```
<rigid_bc type="rigid_fixed">
  <rb>1</rb>
  <Rx_dof>1</Rx_dof>
  <Ry_dof>1</Ry_dof>
  <Rz_dof>1</Rz_dof>
</rigid_bc>
```

3.12.1.2 Rigid_displacement Constraint

The *rigid_displacement* constraint prescribes the value of a rigid displacement degree of freedom. It requires the following parameters.

parameters	Description
rb	The rigid body's material name (1)
dof	The rigid degree of freedom to prescribe. (2)
value	The prescribed value
relative	Defines the value as relative or absolute. (3)

Comments:

1. This is the name attribute assigned to the corresponding rigid body material as defined in the Material section.
2. The values allowed for the *dof* parameter are: x, y, or z.
3. If the relative flag is set, the value is taken relative to the dof value at the start of the step.

The following example prescribes the x-displacement degree of freedom.

```
<rigid_bc type="rigid_displacement">
  <rb>rigid</rb>
  <dof>x</dof>
  <value lc="1">3.14</value>
</rigid_bc>
```

3.12.1.3 Rigid_rotation Constraint

The *rigid_rotation* constraint prescribes the value of a rigid rotational degree of freedom. It requires the following parameters.

parameters	Description
rb	The rigid body's material name (1)
dof	The rigid degree of freedom to prescribe. (2)
value	The prescribed value
relative	Defines the value as relative or absolute. (3)

Comments:

1. This is the name attribute assigned to the corresponding rigid body material as defined in the Material section.
2. The values allowed for the *dof* parameter are: Ru, Rv, or Rw.
3. If the relative flag is set, the value is taken relative to the dof value at the start of the step.

The following example prescribes a rotation around the x-axis.

```
<rigid_bc type="rigid_rotation">
  <rb>rigid</rb>
  <dof>Ru</dof>
  <value lc="1">3.14</value>
</rigid_bc>
```

3.12.2 Rigid Initial Conditions

The *rigid_ic* section defines the initial conditions on rigid bodies. The following rigid initial conditions can be applied.

type	Description
initial_rigid_velocity	Set the initial value for the rigid body velocity.
initial_rigid_angular_velocity	Set the initial value for the rigid body angular velocity.

3.12.2.1 Initial Rigid Velocity

In dynamic analysis, the initial velocity of a rigid body can be set via the *initial_rigid_velocity* rigid constraint. It requires the following parameters.

parameters	Description
rb	The rigid body's material name (1)
value	The initial velocity vector

Comments:

1. This is the "name" attribute assigned to the corresponding rigid body material as defined in the Material section.

The following example defines an initial velocity for a rigid body.

```
<rigid_ic type="initial_rigid_velocity">
  <rb>rigid</rb>
  <value>1,0,0</value>
</rigid_ic>
```

3.12.2.2 Initial Rigid Angular Velocity

In dynamic analysis, the initial angular velocity of a rigid body can be set via the *initial_rigid_angular_velocity* rigid constraint. It requires the following parameters.

parameters	Description
rb	The rigid body's material name. (1)
value	The initial angular velocity vector

Comments:

1. This is the "name" attribute assigned to the corresponding rigid body material as defined in the Material section.

The following example defines an initial angular velocity for a rigid body.

```
<rigid_ic type="initial_rigid_angular_velocity">
  <rb>rigid</rb>
  <value>1,0,0</value>
</rigid_ic>
```

3.12.3 Rigid Loads

The *rigid_load* section can be used to applied loads to a rigid body. The following rigid loads can be defined.

type	Description
rigid_force	Apply a force to the center of the rigid body
rigid_axial_force	An axial force between two rigid bodies.
rigid_follower_force	A force that follows the motion of the rigid body.
rigid_follower_moment	A moment that follows the motion of the rigid body.
rigid_cable	A force applied through a "cable" that connects multiple rigid bodies.

3.12.3.1 Rigid_force Load

The *rigid_force* load applies a load directly to the rigid degree of freedom. It requires the following parameters.

parameters	Description
rb	The rigid body's material name (1)
dof	The rigid degree of freedom to prescribe. (2)
value	The prescribed force value
load_type	Defines the type of load. (3)

Comments:

1. This is the “name” attribute assigned to the corresponding rigid body material as defined in the Material section.
2. The values allowed for the *dof* parameter are: Rx, Ry, Rz.
3. The load_type allows the following values:
   (a) 0 = a load (force/moment) is applied directly to the degree of freedom.
   (b) 1 = a follower load applied is applied to the rigid body. The load is applied in the rigid body’s coordinate system, so it rotates with the rigid body. This only works for Rx, Ry, Rz.
   (c) 2 = target load. The load’s value is ramped up from its initial value at the start of the step, and ramped up linearly to the value specified in the *value* parameter.

The following example applies a force in the x-direction.

```
<rigid_load type="rigid_force">
  <rb>1</rb>
  <dof>Rx</dofs>
  <value lc="1">3.14</value>
</rigid_load>
```

3.12.3.2 Rigid_moment Load

The *rigid_moment* load applies a moment about a partical coordinate axis. It requires the following parameters.

parameters	Description
rb	The rigid body’s material name (1)
dof	The rigid degree of freedom to prescribe. (2)
value	The prescribed moment value
relative	Defines the type of load. (3)

Comments:

1. This is the “name” attribute assigned to the corresponding rigid body material as defined in the Material section.
2. The values allowed for the *dof* parameter are: Ru, Rv, Rw.
3. The relative flag indicates whether the applied value should be absolute or relative to the value at the end of the previous analysis step.

The following example applies a moment about the x-axis.

```
<rigid_load type="rigid_moment">
  <rb>rigid</rb>
  <dof>Ru</dofs>
  <value lc="1">3.14</value>
</rigid_load>
```

3.12.3.3 Rigid_follower_force Load

The *rigid_follower_force* load applies a force in the local rigid body coortinate system. The direction of the force in global coordinates therefore depends on orientation of the rigid body. It requires the following parameters.

parameters	Description
rb	The rigid body's material name (1)
insertion	Location where the force is applied. (2)
force	The vector force value
relative	Relative flag (3)

Comments:

1. This is the "name" attribute assigned to the corresponding rigid body material as defined in the Material section.
2. This is the position in the reference coordinate system of the point where the force is applied.
3. The relative flag indicates whether the applied value should be absolute or relative to the value at the end of the previous analysis step.

The following example applies a moment about the x-axis.

```
<rigid_load type="rigid_follower_force">
  <rb>rigid</rb>
  <insertion>1,2,3</dofs>
  <force>100,0,0</force>
</rigid_load>
```

3.12.3.4 Rigid_follower_moment Load

The *rigid_follower_*moment load applies a moment in the local rigid body coortinate system. The direction of the moment in global coordinates therefore depends on orientation of the rigid body. It requires the following parameters.

parameters	Description
rb	The rigid body's material name (1)
moment	The vector moment value

Comments:

1. This is the "name" attribute assigned to the corresponding rigid body material as defined in the Material section.

The following example applies a moment about the x-axis.

```
<rigid_load type="rigid_follower_moment">
  <rb>rigid</rb>
  <moment>100,0,0</force>
</rigid_load>
```

3.12.3.5 Rigid_cable Load

The *rigid_cable* load emulates the effect of a force applied on one end of a cable that is connected to one or more rigid bodies. It requires the following parameters.

parameters	Description	Defa
force	The scalar force magnitude	0
force_direction	The unit vector defining the force direction	0,0,0
relative	Indicates whether the through-points are defined in global or relative coordinates (1)	1
rigid_cable_point	A point on a rigid body that the cable runs through. (2)	(non

Comments:

1. The *relative* flag indicates whether the location of the cable's via-points are defined in global coordinates (relative=0) or in the local coordinates of the corresponding rigid body (relative=1).

2. One or more rigid cable points must be defined that define the points that the cable runs through. The first point defined is the anchor, where the cable is attached to. The other points define the via points, i.e. a point where the cable runs through. Each cable point is defined by the rigid body that it connects to and a point on the rigid body that the cable runs through. The point's coordinates are either in the global or the local coordinate system of the rigid body, as determined by the *relative* flag.

The following example illustrates a rigid cable setup.

```
<rigid_load type="rigid_cable">
  <force lc="1">100</force>
  <force_direction>1,0,0</force_direction>
  <relative>1</relative>
  <rigid_cable_point>
    <rigid_body_id>rigid1</rigid_body_id>
    <position>0, -1, 1</position>
  </rigid_cable_point>
  <rigid_cable_point>
    <rigid_body_id>rigid2</rigid_body_id>
    <position>0, 1, 1</position>
  </rigid_cable_point>
</rigid_load>
```

3.12.4 Rigid Connectors

The rigid connector section allows users to define different type of connections between two rigid bodies. The specific connector is defined via the *type* attribute.

The rigid connectors fall into two categories. The first category define different joints that contstrain the relative motion of one rigid body with respect to the second one.

Rigid joints produce nonlinear constraints between rigid bodies, which prevent relative motion except along the degrees of freedom of the joint. The term 'rigid' refers to the bodies, not to

the joints. Each rigid joint needs to define two rigid bodies (a and b), a joint origin common to both bodies, and a set of axes that determine the relative orientation of the joint degrees of freedom. These axes define orthonormal basis vectors $\{\mathbf{e}_1^a, \mathbf{e}_2^a, \mathbf{e}_3^a\}$ and $\{\mathbf{e}_1^b, \mathbf{e}_2^b, \mathbf{e}_3^b\}$ on each rigid body, with both bases being coincident, $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$, at the start of the analysis, and given in world coordinates.

The rigid joint nonlinear constraints produce reaction forces and moments that are enforced with penalty parameters and Lagrange multipliers. The penalty parameters, *force_penalty* and *moment_penalty*, may be conceptualized as stiffnesses of linear/torsional springs that prevent relative translations/rotations of the rigid bodies along degrees of freedom that must remain constrained for that joint. The penalty values should be selected based on a rough estimation of the maximum reactions forces and moments acting at these joints, divided by the maximum amount of linear/angular separation that your analysis can tolerate for that joint. Alternatively, set the *force_penalty* and *moment_penalty* parameters to 1 and turn on the *auto_penalty*; this setting will automatically adjust the *force_penalty* and *moment_penalty* to an appropriate value.

The augmented Lagrangian method is used by default, where the joint reaction force and moment are treated as Lagrange multipliers, augmented at each time step by the product of the force/moment penalty parameter and the linear/angular gap. Use the parameter *maxaug* to control the maximum allowable number of augmentations at each time step (*maxaug*=0 produces the penalty method); use a non-zero value for the parameter *minaug* to ensure a minimum number of augmentations. Augmentations will proceed until the relative change in the reaction force/moment magnitude is less than *tolerance*, and/or the linear gap is less than *gaptol*, and/or the angular gap is less than *angtol*. Setting any of these parameters to zero disables that check.

The nonlinear constraints that enforce these joints produce a non-symmetric stiffness matrix. Therefore, when using rigid joints, use the analysis setting for a non-symmetric formulation, see Section 3.3.5.

The following rigid joints can be defined.

type	Description
rigid spherical joint	Connect rigid bodies at point.
rigid revolute joint	Rotation about a single prescribed axis
rigid prismatic joint	Translation along a single prescribed axis.
rigid cylindrical joint	Rotation and translation about prescribed axis.
rigid planar joint	2D plane translation and rotation about normal.
rigid lock	Prevent any relative motion between rigid bodies.

The second category of connectors apply a force on the rigid bodies that is proportional to some relative motion. The term ‘rigid’ refers to the bodies, not to the connectors.

The following rigid connectors can be defined.

type	Description
rigid spring	Applies a spring that connects the two rigid bodies.
rigid damper	Applies a damper based on relative linear velocity.
rigid angular damper	Applies a damper based on relative angular velocity.
rigid contractile force	Constant contractile force between rigid bodies.
rigid planar joint	2D plane translation and rotation about normal.
rigid lock	Prevent any relative motion between rigid bodies.

All of these joints and connectors define two parameters that reference the two rigid bodies.

parameter	Description
body_a	The first rigid body in the connector
body_b	The second rigid body in the connector

In addition, the joints definde above are all based on the Augmented Lagrangian method, and consequently, also share the following parameters.

parameter	Description
laugon	Augmentation flag
tolerance	Specifies the augmentation tolerance
minaug	The minimum number of augmentations.
maxaug	The maximum number of augmentations.
gaptol	Gap tolerance
angtol	Angular separation tolerance
force_penalty	Penalty factor applied to force augmentations.
moment_penalty	Penalty factor applied to moment augmentations.
auto_penalty	Flag to calculate penalty factor automatically.

Additional parameters, unique to a specific connector, are defined in the sections below.

3.12.4.1 Rigid Spherical Joint

A rigid spherical joint connects rigid bodies a and b at a point in space, allowing three degrees of freedom for rotation about that point.

In addition to the shared parameters above, it defines the following parameters.

parameter	Description
joint_origin	The position of the joint.
prescribed_rotation	The prescribed rotation flag.
rotation_x	The x-component of the prescribed rotation.
rotation_y	The y-component of the prescribed rotation.
rotation_z	The z-component of the prescribed rotation.
moment_x	X-component of applied moment
moment_y	Y-component of applied moment
moment_z	Z-component of applied moment

The *tolerance* element defines the augmentation tolerance. That is, when the relative change in the constraint forces and moments (the Lagrange multipliers) are less than this value. The *gaptol* element defines the tolerance for spatial separation of the joint origin on the two bodies (in units of length). Setting either of these elements to zero disables the enforcement of that tolerance. The *force_penalty* parameter (with units of force per length) represents the stiffness that prevents the joint origin on the two bodies from separating. The *body_a* and *body_b* elements are the material numbers of the two rigid bodies. The *joint_origin* element defines the position of the joint (the origin of the basis $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$) in world coordinates at the start of the analysis. Note that this point does not have to be inside or on the surface of either of the two bodies. The *rotation_axis* element defines the orientation of the joint rotation axis in world coordinates at the start of the analysis.

Optionally, the rotation of body b relative to body a may be prescribed using the additional tags

```
<moment_penalty>1e0</moment_penalty>
<prescribed_rotation>1</prescribed_rotation>
<rotation_x lc="1">1</rotation_x>
<rotation_y lc="2">1</rotation_y>
<rotation_z>0</rotation_z>
```

The *prescribed_rotation* element is a flag that indicates that the motion of the joint is prescribed (1 for prescribed, 0 for free). The *rotation_x*, *rotation_y* and *rotation_z* elements specify the components $(\theta_1, \theta_2, \theta_3)$ of rotation (with units of radians), with optional associated load curves. The rotation occurs about the axis directed along $\theta_1\mathbf{e}_1^a + \theta_2\mathbf{e}_2^a + \theta_3\mathbf{e}_3^a$, with a magnitude $\sqrt{\theta_1^2 + \theta_2^2 + \theta_3^2}$. Either all or none of the rotation components must be prescribed, since all rotation components are needed to define a rotation tensor. The *moment_penalty* parameter (with units of moment per radians) represents the torsional stiffness that enforces tracking of the prescribed rotations between the two bodies.

Optionally, moments may be prescribed on body b relative to body a, about the world coordinate axes, using the additional tag

```
<moment_x lc="3">1.e-3</moment_x>
<moment_y lc="4">3.e-3</moment_y>
<moment_z lc="5">2.e-3</moment_z>
```

The *moment* elements specify the components of the moment vector in world coordinates, with optional associated load curves. The *moment* elements should not be used simultaneously with a prescribed rotation.

Example:

```
<rigid_connector type="rigid spherical joint" name="Joint01">
  <tolerance>0.1</tolerance>
  <gaptol>0</gaptol>
  <force_penalty>1e0</force_penalty>
  <auto_penalty>1</auto_penalty>
  <body_a>1</body_a>
  <body_b>2</body_b>
  <joint_origin>0,0,0</joint_origin>
</rigid_connector>
```

3.12.4.2 Rigid Revolute Joint

The rigid revolute joint connector allows the user to connect two rigid bodies a and b at a point in space and allow rotation about a single prescribed axis.

In addition to the shared parameters above, it defines the following parameters.

parameter	Description
joint_origin	The position of the joint.
rotation_axis	The rotation axis.
transverse_axis	The transverse axis.
prescribed_rotation	Prescribed rotation flag
rotation	Prescribed rotation around axis.
moment	Applied moment about axis.

The *tolerance* element defines the augmentation tolerance. That is, when the relative change in the constraint forces and moments (the Lagrange multipliers) are less than this value. The *gaptol* element defines the tolerance for spatial separation of the joint origin on the two bodies (in units of length). The *angtol* element defines the tolerance for angular separation of the joint axis on the two bodies (in units of radians). Setting any of these three elements to zero disables the enforcement of that tolerance. The *force_penalty* parameter (with units of force per length) represents the stiffness that prevents the joint origin on the two bodies from separating. The *moment_penalty* parameter (with units of moment per radians) represents the torsional stiffness that enforces parallelism of the joint axis on the two bodies. The *body_a* and *body_b* elements are the material numbers of the two rigid bodies. The *joint_origin* element defines the position of the joint origin (the origin of the basis $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$) in world coordinates at the start of the analysis. Note that this point does not have to be inside or on the surface of either of the two bodies. The *rotation_axis* element defines the orientation of the joint rotation axis $\mathbf{e}_1$ in world coordinates at the start of the analysis.

Optionally, the rotation of body b relative to body a may be prescribed using the additional tags

```
<prescribed_rotation>1</prescribed_rotation>
<rotation lc="1">3.14159</rotation>
```

The *prescribed_rotation* element is a flag that indicates that the motion of the joint is prescribed (1 for prescribed, 0 for free). The *rotation* element specifies the amount of rotation (with units of radians) with an optional associated load curve.

Optionally, a moment may be prescribed on body b relative to body a, about the joint axis, using the additional tag

```
<moment lc="1">5.e-3</moment>
```

The *moment* element specifies the magnitude of the moment, with an optional associated load curve. The *moment* element should not be used simultaneously with a prescribed rotation.

Example:

```
<rigid_connector type="rigid revolute joint" name="Joint 1">
  <tolerance>0</tolerance>
  <gaptol>1e-4</gaptol>
  <angtol>1e-4</angtol>
  <force_penalty>1e0</force_penalty>
  <moment_penalty>1e0</moment_penalty>
  <auto_penalty>1</auto_penalty>
  <body_a>1</body_a>
  <body_b>3</body_b>
  <joint_origin>-150,0,2</joint_origin>
  <rotation_axis>0,0,1</rotation_axis>
</rigid_connector>
```

3.12.4.3 Rigid Prismatic Joint

The rigid prismatic joint connector allows the user to connect two rigid bodies a and b at a point in space and allow translation along a single prescribed axis.

In addition to the shared parameters above, it defines the following parameters.

parameter	Description
joint_origin	The position of the joint.
translation_axis	The translation axis.
transverse_axis	The transverse axis.
prescribed_translation	Prescribed translation flag
translation	Prescribed translation along axis.
force	Applied force along axis.

The *tolerance* element defines the augmentation tolerance. That is, when the relative change in the constraint forces and moments (the Lagrange multipliers) are less than this value. The *gaptol* element defines the tolerance for spatial separation of the joint origin on the two bodies (in units of length). The *angtol* element defines the tolerance for angular separation of the joint axis on the two bodies (in units of radians). Setting any of these three elements to zero disables the enforcement of that tolerance. The *force_penalty* parameter (with units of force per length) represents the stiffness that prevents the joint origin on the two bodies from separating. The *moment_penalty* parameter (with units of moment per radians) represents the torsional stiffness that enforces parallelism of the joint axis on the two bodies. The *body_a* and *body_b* elements are the material numbers of the two rigid bodies. The *joint_origin* element defines the position of the joint (the origin of the basis $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$) in world coordinates at the start of the analysis. Note that this point does not have to be inside or on the surface of either of the two bodies. The *translation_axis* element defines the orientation $\mathbf{e}_1$ of the joint translation axis in world coordinates at the start of the analysis.

Optionally, the translation of body b relative to body a may be prescribed using the additional tags

```
<prescribed_translation>1</prescribed_translation>
<translation lc="1">5.0</translation>
```

The *prescribed_translation* element is a flag that indicates that the motion of the joint is prescribed (1 for prescribed, 0 for free). The *translation* element specifies the amount of translation (with units of length) with an optional associated load curve.

Optionally, a force may be prescribed on body b relative to body a, along the joint axis using the additional tag

```
<force lc="1">5.e-3</force>
```

The *force* element specifies the magnitude of the force, with an optional associated load curve. The *force* element should not be used simultaneously with a prescribed translation.

Example:

```
<rigid_connector type="rigid prismatic joint" name="Joint02 ">
  <tolerance>0</tolerance>
  <gaptol>1e-4</gaptol>
  <angtol>1e-4</angtol>
  <force_penalty>1e0</force_penalty>
  <moment_penalty>1e0</moment_penalty>
  <auto_penalty>1</auto_penalty>
  <body_a>4</body_a>
  <body_b>1</body_b>
  <joint_origin>-150,0,2</joint_origin>
  <translation_axis>1,0,0</translation_axis>
  <prescribed_translation>150</prescribed_translation>
  <translation lc="1">1</translation>
</rigid_connector>
```

3.12.4.4 Rigid Cylindrical Joint

A rigid cylindrical joint connects rigid bodies a and b at a point in space, allowing one degree of freedom for rotation about an axis through that point, and another degree of freedom for translation along that axis.

In addition to the shared parameters above, it defines the following parameters.

parameter	**Description**
joint_origin	The position of the joint.
joint_axis	The joint axis.
transverse_axis	The transverse axis.
prescribed_translation	Prescribed translation flag
translation	Prescribed translation along axis.
force	Applied force along axis.
prescribed_rotation	Prescribed rotation flag
rotation	Prescribed rotation around axis.
moment	Applied moment about axis.

The *tolerance* element defines the augmentation tolerance. That is, when the relative change in the constraint forces and moments (the Lagrange multipliers) are less than this value. The *gaptol* element defines the tolerance for spatial separation of the joint origin on the two bodies (in units of length). The *angtol* element defines the tolerance for angular separation of the joint axis on the two bodies (in units of radians). Setting any of these three elements to zero disables the enforcement of that tolerance. The *force_penalty* parameter (with units of force per length) represents the stiffness that prevents the joint origin on the two bodies from separating. The *moment_penalty* parameter (with units of moment per radians) represents the torsional stiffness that enforces parallelism of the joint axis on the two bodies. The *body_a* and *body_b* elements are the material numbers of the two rigid bodies. The *joint_origin* element defines the position of the joint (the origin of the basis $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$) in world coordinates at the start of the analysis. Note that this point does not have to be inside or on the surface of either of the two bodies. The *joint_axis* element defines the orientation $\mathbf{e}_1$ of the joint rotation and translation axis in world coordinates at the start of the analysis.

Optionally, the rotation of body b relative to body a may be prescribed using the additional tags

```
<prescribed_rotation>1</prescribed_rotation>
<rotation lc="1">1.570796327</rotation>
```

The *prescribed_rotation* element is a flag that indicates that the motion of the joint is prescribed (1 for prescribed, 0 for free). The *rotation* element specifies the amount of rotation (with units of radians) with an optional associated load curve.

Optionally, the translation of body b relative to body a may be prescribed using the additional tags

```
<prescribed_translation>1</prescribed_translation>
<translation lc="2">10.0</translation>
```

Optionally, a moment may be prescribed about the joint axis using the additional tag

```
<moment lc="1">5.e-3</moment>
```

The *moment* element specifies the magnitude of the moment, with an optional associated load curve. The *moment* element should not be used simultaneously with a prescribed rotation.

Optionally, a force may be prescribed along the joint axis using the additional tag

```
<force lc="1">2.0</force>
```

The *force* element specifies the magnitude of the force, with an optional associated load curve. The *force* element should not be used simultaneously with a prescribed translation.

Example:

```
<rigid_connector type="rigid cylindrical joint" name="Joint03">
  <tolerance>0</tolerance>
  <gaptol>1e-4</gaptol>
  <angtol>1e-4</angtol>
  <force_penalty>1e0</force_penalty>
```

```
    <moment_penalty>1e0</moment_penalty>
    <auto_penalty>1</auto_penalty>
    <body_a>1</body_a>
    <body_b>2</body_b>
    <joint_origin>0,0,0</joint_origin>
    <joint_axis>0,0,1</joint_axis>
  </rigid_connector>
```

3.12.4.5 Rigid Planar Joint

A rigid planar joint connects rigid bodies a and b, allowing one degree of freedom for rotation about the axis $\mathbf{e}_1$ through that point, and two degrees of freedom for translations in the plane perpendicular to that axis, along $\mathbf{e}_2^a$ and $\mathbf{e}_3^a$.

In addition to the shared parameters above, it defines the following parameters.

parameter	**Description**
joint_origin	The position of the joint.
rotation_axis	The joint axis.
translation_axis_1	The first translation axis.
prescribed_rotation	Prescribed rotation flag
rotation	Prescribed rotation around axis.
prescribed_translation_1	Prescribed translation 1 flag
translation_1	Prescribed translation along axis 1
prescribed_translation_2	Prescribed translation 2 flag
translation_2	Prescribed translation along axis 2

The *tolerance* element defines the augmentation tolerance. That is, when the relative change in the constraint forces and moments (the Lagrange multipliers) are less than this value. The *gaptol* element defines the tolerance for spatial separation of the joint origin on the two bodies (in units of length). The *angtol* element defines the tolerance for angular separation of the joint axes on the two bodies (in units of radians). Setting any of these three elements to zero disables the enforcement of that tolerance. The *force_penalty* parameter (with units of force per length) represents the stiffness that prevents the joint origin on the two bodies from separating along the rotation axis. The *moment_penalty* parameter (with units of moment per radians) represents the torsional stiffness that enforces parallelism of the joint rotation axis on the two bodies. The *body_a* and *body_b* elements are the material numbers of the two rigid bodies. The *joint_origin* element defines the position of the joint (the origin of the basis $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$) in world coordinates at the start of the analysis. Note that this point does not have to be inside or on the surface of either of the two bodies. The *rotation_axis* element defines the orientation of the joint rotation axis $\mathbf{e}_1$ in world coordinates at the start of the analysis. The *translation_axis_1* element defines the orientation of the joint translation axis $\mathbf{e}_2^a$ in the plane perpendicular to the joint rotation axis, in world coordinates at the start of the analysis.

Optionally, the rotation of body b relative to body a may be prescribed using the additional tags

```
<prescribed_rotation>1</prescribed_rotation>
<rotation lc="1">1.570796327</rotation>
```

The *prescribed_rotation* element is a flag that indicates that the motion of the joint is prescribed (1 for prescribed, 0 for free). The *rotation* element specifies the amount of rotation (with units of radians) with an optional associated load curve.

Optionally, the translation of body b relative to body a may be prescribed along $\mathbf{e}_2^a$ using the additional tags

```
<prescribed_translation_1>1</prescribed_translation_1>
<translation_1 lc="2">10.0</translation_1>
```

Optionally, the translation of body b relative to body a may be prescribed along $\mathbf{e}_3^a$ using the additional tags

```
<prescribed_translation_2>1</prescribed_translation_2>
<translation_2 lc="2">10.0</translation_2>
```

Example:

```
<rigid_connector type="rigid planar joint" name="Joint01">
  <tolerance>0</tolerance>
  <gaptol>1e-4</gaptol>
  <angtol>1e-4</angtol>
  <force_penalty>1e0</force_penalty>
  <moment_penalty>1e0</moment_penalty>
  <auto_penalty>1</auto_penalty>
  <body_a>1</body_a>
  <body_b>2</body_b>
  <joint_origin>0,0,0</joint_origin>
  <rotation_axis>0,-0.5,0.8660254</rotation_axis>
  <translation_axis_1>1,0,0</translation_axis_1>
</rigid_connector>
```

3.12.4.6 Rigid Lock Joint

A rigid lock connects rigid bodies a and b, preventing relative motion between them. It requires the specification of a joint origin and two orthgonal axes $\mathbf{e}_1$ and $\mathbf{e}_2$.

In addition to the shared parameters above, it defines the following parameters.

parameter	Description
joint_origin	The position of the joint.
first_axis	The first axis.
second_axis	The second axis.

The *tolerance* element defines the augmentation tolerance. That is, when the relative change in the constraint forces and moments (the Lagrange multipliers) are less than this value. The *gaptol* element defines the tolerance for spatial separation of the joint origin on the two bodies (in units of length). The *angtol* element defines the tolerance for angular separation of the joint axes on the two bodies (in units of radians). Setting any of these three elements to zero disables the enforcement of that tolerance. The *force_penalty* parameter (with units of force per length) represents the stiffness that prevents the joint origin on the two bodies from separating along the rotation axis. The *moment_penalty* parameter (with units of moment per radians) represents the torsional stiffness that enforces parallelism of the joint rotation axis on the two bodies. The *body_a* and *body_b* elements are the material numbers of the two rigid bodies. The *joint_origin* element defines the position of the joint (the origin of the basis $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$) in world coordinates at the start of the analysis. Note that this point does not have to be inside or on the surface of either of the two bodies. The *first_axis* element defines the orientation of the axis $\mathbf{e}_1$ in world coordinates at the start of the analysis. The *second_axis* element defines the orientation of the second axis $\mathbf{e}_2$ in world coordinates at the start of the analysis.

Example:

```
<rigid_connector type="rigid lock" name="Joint01">
  <tolerance>0</tolerance>
  <gaptol>1e-4</gaptol>
  <angtol>1e-4</angtol>
  <force_penalty>1e0</force_penalty>
  <moment_penalty>1e0</moment_penalty>
  <auto_penalty>1</auto_penalty>
  <body_a>1</body_a>
  <body_b>2</body_b>
  <joint_origin>0,0,0</joint_origin>
  <first_axis>1,0,0</first_axis>
  <second_axis>0,1,0</second_axis>
</rigid_connector>
```

3.12.4.7 Rigid Spring

The rigid spring applies a linear spring that connects two rigid bodies a and b at arbitrary points (not necessarily nodes).

In addition to the shared parameters above, it defines the following parameters.

parameter	Description
k	spring constant (in units of force per length)
insertion_a	Insertion point of the spring on body a
insertion_b	Insection point of the spring on body b
free_length	The length of the unloaded spring (1)

Comments:

1. If the free_length parameter is zero, then the initial length of the spring, as defined by the two insertion points, is taken as the free length.

Example:

```
<rigid_connector type="rigid spring">
  <body_a>1</body_a>
  <body_b>2</body_b>
  <insertion_a>0,0,1</insertion_a>
  <insertion_b>0,0,3</insertion_b>
  <k>1</k>
</rigid_connector>
```

3.12.4.8 Rigid Damper

The rigid damper applies a linear damper that connects two rigid bodies a and b at arbitrary points (not necessarily nodes).

In addition to the shared parameters above, it defines the following parameters.

parameter	Description
c	Damping constant
insertion_a	Insertion point of the spring on body a
insertion_b	Insection point of the spring on body b

Example:

```
<rigid_connector type="rigid damper">
  <body_a>1</body_a>
  <body_b>2</body_b>
  <insertion_a>0,0,1</insertion_a>
  <insertion_b>0,0,3</insertion_b>
  <c>1e-7</c>
</rigid_connector>
```

3.12.4.9 Rigid Angular Damper

The rigid angular damper applies an angular damper that connects two rigid bodies a and b.

In addition to the shared parameters above, it defines the following parameters.

parameter	Description
c	Angular damping constant

Example:

```
<rigid_connector type="rigid angular damper">
  <body_a>1</body_a>
  <body_b>2</body_b>
  <c>1e-3</c>
</rigid_connector>
```

3.12.4.10 Rigid Contractile Force

The rigid contractile force applies an actuator force between arbitrary points (not necessarily nodes) on two rigid bodies a and b.

In addition to the shared parameters above, it defines the following parameters.

parameter	Description
f0	The applied force (1)
insertion_a	Insertion point of the spring on body a
insertion_b	Insection point of the spring on body b

Comments:

1. The f0 parameter represents the actuator force (positive for contraction). Optionally, it may be associated with a load curve.

Example:

```
<rigid_connector type="rigid contractile force">
  <body_a>1</body_a>
  <body_b>2</body_b>
  <insertion_a>0,0,1</insertion_a>
  <insertion_b>0,0,3</insertion_b>
  <f0 lc="1">1</f0>
</rigid_connector>
```

3.13 Loads Section

The *Loads* section defines all nodal, edge, surface, and body loads that can be applied to the model.

3.13.1 Nodal Loads

Nodal loads are applied by the *nodal_load* element. The nodal load is defined by the *type* attribute. The following table lists the available nodal loads.

type	Description
nodal_load	A load that is added to the corresponding's dofs in the global force vector.
nodal_force	A force applied to each node of a node set

3.13.1.1 nodal_load

The nodal_load adds a value directly to the corresponding degrees of freedom in the global external load vector. When the loads are applied to displacement degrees of freedom, the forces always point in the same direction and do not deform with the geometry (i.e. they are non-follower forces). For other degrees of freedom they define a constant normal flux.

```
<nodal_load type="nodal_load" node_set="set1">
  <dof>x</dof>
  <scale lc="1">1.0</scale>
</nodal_load>
```

The *dof* parameter defins the degree of freedom. The following values are allowed:

x apply force in x-direction

y apply force in y-direction

z apply force in z-direction

p apply normal volumetric fluid flow rate

cn apply normal molar solute flow rate

t apply normal heat flux (heat transfer analysis)

For solutes, replace "n" with the solute id from the global solute table (Section 3.4.2); for example, "c2".

An optional *loadcurve* can be specified for the *scale* parameter with the *lc* attribute.

3.13.1.2 nodal_force

This nodal load defines a force that is applied to each node in the node set. The *value* parameter sets the force vector. An optional load curve can be defined to scale the vector.

```
<nodal_load type="nodal_force" node_set="set1">
  <value lc="1">1,0,0</scale>
</nodal_load>
```

3.13.2 Surface Loads

A surface load can be applied using the *surface_load* element. This element takes two attributes, namely *type*, which defines the type of surface load that will be applied, and *surface* which defines the surface that this load will be applied to.

3.13.2.1 Pressure Load

Pressure loads are applied to the surface of the geometry and are defined by the *surface_load* element with the type attribute set to *pressure*. These pressure forces are also known as *follower forces*; they change direction as the body is deformed and, in this case, are always oriented along the local surface normal. The sign convention is so that a positive pressure will act opposite to the normal, so it will compress the material. The surface that the load is applied to is specified with the *surface* attribute.The surface must be defined in the *Mesh* section.

parameter	**Description**	default
pressure	The applied pressure values (1)	0 [**P**]
symmetric_stiffness	symmetric stiffness flag (2)	1
linear	linear flag (3)	0
shell_bottom	shell bottom flag (4)	0

Comments:

1. The *pressure* element defines the pressure value [**P**]. The optional parameter *lc* defines a loadcurve for the pressure evolution. If *lc* is not defined a constant pressure is applied.

2. The optional *symmetric_stiffness* parameter (0 or 1) determines whether to use the exact form of the stiffness matrix for this surface load (0=non-symmetric), or the symmetric form (1=default). The non-symmetric form is numerically more robust, but it may also require using a globally non-symmetric stiffness matrix (see Section 3.3.5).

3. If the optional *linear* parameter is set to 1, a constant, deformation independent pressure load is applied. This is not recommend for large deformations.

4. When the *shell_bottom* flag is set to 1, the surface is assumed to be a shell surface and the pressure is applied to the bottom part of the shell surface.

Example:

```
<surface_load type="pressure" surface="surface1">
  <pressure lc="1">1.0</pressure>
  <symmetric_stiffness>0</symmetric_stiffness>
</surface_load>
```

3.13.2.2 Traction Load

A traction load applies a traction to a surface. The direction of the traction remains unchanged as the mesh deforms.

```
<surface_load type="traction" surface="surface1">
  <scale lc="1">1.0</scale>
  <traction>0,0,1</traction>
</surface_load>
```

The traction vector is determined by two quantities. The direction and magnitude is defined by the *traction* element. In addition, the magnitude can be scaled using the *scale* element. An optional load curve can be defined for the *scale* element using the *lc* attribute. This allows the traction load to become time dependent. If the *lc* attribute is omitted a constant traction load is applied.

3.13.2.3 Surface Force

A surface force applies an equivalent uniform traction load to a surface. The direction of the force remains unchanged as the mesh deforms.

```
<surface_load type="force" surface="surface1">
  <scale lc="1">1.0</scale>
  <force>0,0,1</force>
</surface_load>
```

The force vector is determined by two quantities. The direction and magnitude is defined by the *force* element. In addition, the magnitude can be scaled using the *scale* element. An optional load curve can be defined for the *scale* element using the *lc* attribute. This allows the surface force to become time dependent. If the *lc* attribute is omitted a constant force is applied.

3.13.2.4 Bearing Load

A bearing load can be applied on a cylindrical surface that represents a bearing journal, or the portion of a shaft that is supported by a bearing journal. This load gets prescribed as a non-uniform compressive pressure distribution on that surface, representing the radial component of the bearing load. (The axial component of a bearing load may be prescribed using the surface force presented in Section 3.13.2.3.) The non-uniform pressure distribution may be sinusoidal or parabolic. It gets distributed over a ±90 degree arc relative to the loading direction, which is specified as a force vector. The user is expected to specify the bearing force in a plane perpendicular to the axis of the cylindrical bearing surface. Otherwise, only the projection of the bearing force onto that plane will be prescribed as the radial bearing load. For example, a bearing load may be specified as

```
<surface_load type="bearing load" surface="BearingSurface1">
  <scale lc="1">1000.0</scale>
  <force>0.707107,0.707107,0</force>
  <profile>1</profile>
</surface_load>
```

In this example the bearing load is in the xy-plane, oriented at 45 degrees from the x-axis. The surface *BearingSurface1* should be a cylinder with its axis along the z-direction. The pressure distribution is parabolic (profile=1), and the alternative option is sinusoidal (profile=0). In general, a parabolic distribution produces a higher (often more realistic) central peak pressure on the bearing surface. An optional loadcurve can be associated with the *scale* as shown in this example, to scale the prescribed bearing force as a function of time. Alternatively, the user may assign a loadcurve to the *force*.

Additional options are available for a bearing load, including *symmetric_stiffness*, *linear* and *shell_bottom* as outlined in Section 3.13.2.1. When the *linear* flag is set to false, the pressure prescribed on the bearing surface is adjusted to account for the change in area and orientation of each face, caused by the finite deformation of the bearing surface. However, the bearing radial force direction remains unchanged with deformation. It is recommended to set the *symmetric_stiffness* flag to false when performing a finite deformation analysis.

The bearing load is always prescribed as a compressive load on the selected bearing surface. Therefore, it is recommended to select a complete cylindrical surface on which the bearing load is prescribed; the solver automatically determines which faces of the selected surface must be loaded in compression and which of those have zero pressure prescribed on them, based on the vectorial orientation of the bearing *force*. If the user selects a semi-cylindrical surface as the bearing surface, it becomes the user's responsibility to ensure that the prescribed bearing force

bisects the 180 degree arc of that surface. Otherwise, the analysis may produce unexpected results, due to unevenness of the pressure distribution about the force's line of action. If the user selects a surface with a smaller arc than ±90 degrees, the pressure will not reduce to zero at the straight edges of the surface, though the pressure will be scaled to produce the prescribed force magnitude along the bearing force direction.

3.13.2.5 Mixture Normal Traction

This section applies to biphasic, biphasic-solute, triphasic and multiphasic analyses. In a mixture of intrinsically incompressible solid and fluid constituents, the formulation adopted in FEBio implies that the total traction is a natural boundary condition (*FEBio Theory Manual*). If this boundary condition is not explicitly prescribed, the code automatically assumes that it is equal to zero. Therefore, boundaries of mixtures are traction-free by default.

The *mixture traction* $\mathbf{t}$ is the traction vector corresponding to the mixture (or total) stress $\boldsymbol{\sigma}$; thus $\mathbf{t} = \boldsymbol{\sigma} \cdot \mathbf{n}$, where $\mathbf{n}$ is the outward unit normal to the boundary surface. Since $\boldsymbol{\sigma} = -p\mathbf{I} + \boldsymbol{\sigma}^e$, where p is the fluid pressure and $\boldsymbol{\sigma}^e$ is the *effective stress* resulting from strains in the solid matrix, it is also possible to represent the total traction as $\mathbf{t} = -p\mathbf{n} + \mathbf{t}^e$, where $\mathbf{t}^e = \boldsymbol{\sigma}^e \cdot \mathbf{n}$ is the *effective traction*. Currently, FEBio allows the user to specify only the normal component of the traction, either $t_n = \mathbf{t} \cdot \mathbf{n}$ (the normal component of the mixture traction) or $t_n^e = \mathbf{t}^e \cdot \mathbf{n}$ (the normal component of the effective traction):

A mixture normal traction is defined by the *surface_load* element using *normal_traction* for the type attribute.

```
<surface_load type="normal_traction" surface="surface1">
  <traction [lc="1"]>1.0</traction>
  <effective>1</effective>
  <linear>0</linear>
</surface_load>
```

The *traction* element defines the magnitude of the traction force. The optional attribute *lc* defines a loadcurve that controls the time dependency of the traction force magnitude. If omitted a constant traction is applied.

The *effective* element defines whether the traction is applied as an effective traction or a total mixture traction.

The *linear* element defines whether the traction remains normal to the deformed surface or the reference surface. If set to true the traction remains normal to the reference surface. When false it defines a follower force that remains normal to the deformed surface.

The *surface* element defines the surface to which the traction is applied. It consists of child elements defining the individual surface facets.

Unlike the mixture and effective traction, the fluid pressure p is a nodal variable (see Section 3.11.1). There may be common situations where the user must apply a combination of related fluid pressure and traction boundary conditions. For example, if a biphasic surface is subjected to a non-zero fluid pressure p_0, the corresponding boundary conditions are $p = p_0$ and $t_n = -p_0$ (or $t_n^e = 0$). In FEBio, both boundary conditions must be applied. For example:

```
<Boundary>
  <prescribed bc="p" node_set="set1">
    <scale lc="1">1.0</scale>
```

```
  </prescribed>
  <surface_load type="normal_traction" surface="surf1">
    <traction lc="1">-1.0</traction>
    <effective>0</effective>
    <linear>0</linear>
  </surface_load>
</Boundary>
<LoadData>
  <loadcontroller id="1" type="loadcurve">
    <points>
      <loadpoint>0,0</loadpoint>
      <loadpoint>1,2.0</loadpoint>
    </points>
  </loadcurve>
</LoadData>
```

3.13.2.6 Fluid Flux

In a biphasic mixture of intrinsically incompressible solid and fluid constituents, the $\mathbf{u}-p$ formulation adopted in FEBio implies that the normal component of the relative fluid flux is a natural boundary condition. If this boundary condition is not explicitly prescribed, the code automatically assumes that it is equal to zero. Therefore, biphasic boundaries are impermeable by default. (To implement a free-draining boundary, the fluid pressure nodal degrees of freedom should be set to zero.)

The flux of fluid relative to the solid matrix is given by the vector $\mathbf{w}$. Since viscosity is not explicitly modeled in a biphasic material, the tangential component of $\mathbf{w}$ on a boundary surface may not be prescribed. Only the normal component of the relative fluid flux, $w_n = \mathbf{w}\cdot\mathbf{n}$, represents a natural boundary condition. To prescribe a value for w_n on a surface, use:

```
<surface_load type="fluidflux" surface="surf1">
  <flux lc="1">1.0</flux>
  <linear>0</linear>
  <mixture>1</mixture>
</fluidflux>
```

The *flux* parameter defines the flux that will be applied to the surface. The optional parameter *lc* defines a loadcurve for the normal flux evolution. If omitted a constant fluid flux is applied.

When *linear* is set to zero (default) it means that the flux matches the prescribed value even if the surface on which it is applied changes in area as it deforms. Therefore, the net volumetric flow rate across the surface changes with changes in area. This type of boundary condition is useful if the fluid flux is known in the current configuration.

When *linear* is set to non-zero it means that the prescribed flux produces a volumetric flow rate based on the undeformed surface area in the reference configuration. Therefore, the flux in the current configuration does not match the prescribed value. This type of boundary condition is useful if the net volumetric flow rate across the surface is known. For example: Let Q be the known volumetric flow rate, let A_0 be the surface area in the reference configuration (a constant). Using *"linear"* means that the user prescribes Q/A_0 as the flux boundary condition. (However, regardless of the *type*, the fluid flux saved in the output file has a normal component equal to Q/A, where $A =$area in current configuration.)

Prescribing w_n on a free surface works only if the nodal displacements of the corresponding faces are also prescribed. If the nodal displacements are not known a priori, the proper boundary condition calls for prescribing the normal component of the mixture velocity, $v_n = (\mathbf{v}^s + \mathbf{w}) \cdot \mathbf{n}$. To prescribe the value of v_n on a surface, use

```
<surface_load type="fluidflux" surface="surf1">
  <flux lc="1">1.0</flux>
  <linear>0</linear>
  <mixture>1</mixture>
</surface_load>
```

For example, this boundary condition may be used when modeling a permeation problem through a biphasic material, when the upstream fluid velocity is prescribed, $v_n = v_0$. If the upstream face is free, the companion boundary condition would be to let $t_n^e = 0$ on that face as well.

3.13.2.7 Multiphasic Solute Flux

In biphasic-solute and multiphasic analyses the molar flux of a solute ι, given by the vector $\mathbf{j}^\iota = \varphi^w c^\iota (\mathbf{v}^\iota - \mathbf{v}^s)$, is evaluated relative to the solid matrix; it includes a diffusive component as well as a convective component contributed by the fluid flux $\mathbf{w}$ relative to the solid, see eq.(4.11.4). Since solute viscosity is not explicitly modeled, the tangential component of $\mathbf{j}^\iota$ on a boundary surface may not be prescribed. Only the normal component of the relative solute flux, $j_n^\iota = \mathbf{j}^\iota \cdot \mathbf{n}$, represents a natural boundary condition. Since a multiphasic mixture may contain electrically charged solutes, the solute flux that must be prescribed as a boundary condition is the effective solute flux $\tilde{j}_n^\iota = j_n^\iota + \sum_\gamma z^\gamma j_n^\gamma$, as explained in Section 4.12.1.4. To prescribe a value for $\tilde{j}_n^\iota$ on a surface, use:

```
<surface_load type="soluteflux" surface="surface1">
  <flux lc="1">1.0</flux>
  <linear>0</linear>
  <solute_id>solute_name</solute_id>
  <shell_bottom>0</shell_bottom>
</surface_load>
```

The parameter *solute_id* specifies to which solute this flux condition applies, referencing the corresponding list of solute names in the Globals section (Section 3.4.2).

The *flux* element defines the flux magnitude. The optional parameter *lc* defines a loadcurve for the normal flux evolution. If omitted a constant flux is applied.

When *linear* is set to 0 (default) it means that the flux matches the prescribed value even if the surface on which it is applied changes in area as it deforms. Therefore, the net molar flow rate across the surface changes with changes in area. This type of boundary condition is useful if the solute molar flux is known in the current configuration.

When *linear* is set to non-zero it means that the prescribed flux produces a molar flow rate based on the undeformed surface area in the reference configuration. Therefore, the flux in the current configuration does not match the prescribed value. This type of boundary condition is useful if the net molar flow rate across the surface is known. For example: Let Q be the known molar flow rate (in units of moles per time [**n**/**t**]), let A_0 be the surface area in the reference configuration (a constant). Using *linear* means that the user prescribes Q/A_0 as the flux boundary condition (in

units of moles per area per time [**n**/**L**^{2}·**t**]). However, regardless of the *type*, the solute molar flux saved in the output file has a normal component equal to Q/A, where A =area in current configuration. The *shell_bottom* tag should be set to 1 (true) when prescribing the flux on the bottom surface of a multiphasic shell.

3.13.2.8 Solute Flux

In fluid-solutes analyses the molar flux of solute ι, given by the vector $\mathbf{j}^\iota = c^\iota\left(\mathbf{v}^\iota - \mathbf{v}^f\right)$, is evaluated relative to the solvent (fluid); it only includes a diffusive contribution relative to the solvent. Since solute viscosity is not explicitly modeled, the tangential component of $\mathbf{j}^\iota$ on a boundary surface may not be prescribed. Only the normal component of the relative solute flux, $j_n^\iota = \mathbf{j}^\iota \cdot \mathbf{n}$, represents a natural boundary condition. Since a fluid-solutes mixture may contain electrically charged solutes, the solute flux that must be prescribed as a boundary condition is the effective solute flux $\tilde{j}_n^\iota = j_n^\iota + \sum_\gamma z^\gamma j_n^\gamma$, such as that in eq.(8.8.1). To prescribe a value for $\tilde{j}_n^\iota$ on a surface, use:

```
<surface_load type="solute flux" surface="surface1">
  <flux lc="1">1.0</flux>
  <solute_id>solute_name</solute_id>
</surface_load>
```

The parameter *solute_id* specifies to which solute this flux condition applies, referencing the corresponding list of solute names in the Globals section (Section 3.4.2).

The *flux* element defines the flux magnitude. The optional parameter *lc* defines a loadcurve for the normal flux evolution. If omitted a constant flux is applied.

3.13.2.9 Multiphasic Solute Natural Flux

This surface load can be prescribed on a multiphasic boundary where a natural solute flux is desired. On this boundary, the solute is convected with the solvent according to $j_n = cw_n$. The implication of this surface load is that there is no diffusive hindrance to solute transport on the external side of the boundary. Here, c is the average solute concentration and $\mathbf{w}$ is the average solvent volumetric flux in the finite element connected to the boundary face on which this surface load is prescribed. The normal solvent flux w_n is evaluated as $\mathbf{w} \cdot \mathbf{n}$ using the normal $\mathbf{n}$ on the boundary. There are no user-specified parameters in this surface load (other than specifying the solute for which it applies, and whether the boundary is the bottom of a shell domain). To prescribe a solute natural flux on a surface, use:

```
<surface_load type="solute natural flux" surface="surface1">
  <solute_id>solute_name</solute_id>
  <shell_bottom>0</shell_bottom>
  <update>0</update>
</surface_load>
```

The parameter *solute_id* specifies to which solute this flux condition applies, referencing the corresponding list of solute names in the Globals section (Section 3.4.2). The parameter *update* is a boolean flag (0=*false*, 1=*true*, default is *false*) which, when set to *true*, initializes the effective solute concentration of surface nodes to the average value of concentrations within the underlying elements, at each iteration. You may set this flag to *true* when the iterative numerical solution fails to converge. This surface load is only needed for biphasic-solute and multiphasic analyses, since $\tilde{j}_n^\iota = 0$ is the natural boundary condition for fluid-solutes analyses.

3.13.2.10 Heat Flux

A heat flux can be defined for heat transfer analyses using the *surface_load* element with type set to *heatflux* element.

```
<surface_load type="heatflux" surface="surface1">
  <flux lc="1">1.0</flux>
</surface_load>
```

The heat flux element takes two parameters, namely *flux* which defines the flux that will be applied. It has an optional *lc* attribute which defines the load curve for this parameter. If omitted, a constant heat flux will be applied. The other (optional) parameter is the *value* parameter which can be used to define a different flux value for each surface facet.

3.13.2.11 Convective Heat Flux

A convective heat flux can be defined to a surface that is cooled (or heated) by exposure to an ambient temperature.

```
<surface_load type="convective_heatflux" surface="surf1">
  <Ta lc="1">1.0</Ta>
  <hc>1.0</hc>
</surface_load>
```

The *hc* parameter defines the heat transfer coefficient. The ambient temperature is defined by the *Ta* parameter. It takes an optional load curve defined through the *lc* attribute. The *value* parameter with the *surface_data* attribute can be defined to define a different temperature value for each surface facet (Section 3.8.1).

3.13.2.12 Fluid Traction

In a fluid or fluid-FSI analysis, the Cauchy stress is given by $\boldsymbol{\sigma} = -p\mathbf{I} + \boldsymbol{\tau}$, where p is the elastic pressure and $\boldsymbol{\tau}$ is the viscous stress. The traction on a fluid surface with outward normal $\mathbf{n}$ is given by $\mathbf{t} = \boldsymbol{\sigma} \cdot \mathbf{n} = -p\mathbf{n} + \mathbf{t}^{\tau}$, where $\mathbf{t}^{\tau} = \boldsymbol{\tau} \cdot \mathbf{n}$ is the viscous component of the traction. The *fluid traction* surface load prescribes $\mathbf{t}^{\tau}$ on a boundary surface.

```
<surface_load type="fluid traction" surface="surface1">
  <scale lc="1">1.0</scale>
  <traction>0,0,1</traction>
</surface_load>
```

3.13.2.13 Fluid Pressure

In a fluid or fluid-FSI analysis, the Cauchy stress is given by $\boldsymbol{\sigma} = -p\mathbf{I} + \boldsymbol{\tau}$, where p is the elastic pressure and $\boldsymbol{\tau}$ is the viscous stress. The *fluid pressure* surface load prescribes p on a boundary surface.

```
<surface_load type="fluid pressure" surface="surface1">
  <pressure>0.05</pressure>
</surface_load>
```

This boundary condition internally solves for the fluid dilatation e^f for the user-prescribed pressure p, using the user-selected equation of state from the list given in eq.(4.16.2). Then $e^f$is prescribed as an essential boundary condition on the selected surface.

3.13.2.14 Fluid Normal Traction

In a fluid or fluid-FSI analysis, the normal component $t_n^\tau = \mathbf{t}^\tau \cdot \mathbf{n}$ of the viscous traction $\mathbf{t}^\tau$ (Section 3.13.2.12) may be prescribed on a boundary surface in a fluid analysis. The *fluid normal traction* surface load prescribes t_n^τ on a boundary surface.

```
<surface_load type="fluid normal traction" surface="surface1">
  <scale lc="1">1.0</scale>
  <traction>1</traction>
</surface_load>
```

3.13.2.15 Fluid Backflow Stabilization

In a fluid or fluid-FSI analysis, this boundary condition prescribes the normal component of the viscous traction (Section 3.13.2.12), $t_n^\tau = \mathbf{t}^\tau \cdot \mathbf{n}$, such that it opposes backflow, i.e., when $v_n = \mathbf{v} \cdot \mathbf{n}$ is negative. Specifically, $t_n^\tau = \beta \rho_r v_n^2$ when $v_n < 0$ and 0 otherwise, where β is a unitless user-defined parameter (typical values may range from 0 to 1) and ρ_r is the referential fluid density.

```
<surface_load type="fluid backflow stabilization" surface="surface1">
<beta lc="1">1</beta>
</surface_load>
```

3.13.2.16 Fluid Tangential Stabilization

In a fluid or fluid-FSI analysis, this boundary condition prescribes the tangential component of the viscous traction (Section 3.13.2.12), $\mathbf{t}_t^\tau = (\mathbf{I} - \mathbf{n} \otimes \mathbf{n}) \cdot \mathbf{t}^\tau$, such that it opposes the tangential fluid flow $\mathbf{v}_t = (\mathbf{I} - \mathbf{n} \otimes \mathbf{n}) \cdot \mathbf{v}$ on the selected boundary surface, with $\mathbf{n}$ representing the outward normal to this surface. Specifically, $\mathbf{t}_t^\tau = -\beta \rho_r |\mathbf{v}_t| \mathbf{v}_t$, where β is a unitless user-defined parameter (typical values may range from 0 to 1) and ρ_r is the referential fluid density.

```
<surface_load type="fluid tangential stabilization" surface="surface1">
<beta lc="1">1</beta>
</surface_load>
```

3.13.2.17 Fluid Normal Velocity

In a fluid or fluid-FSI analysis the fluid velocity relative to the mesh, $\mathbf{w}$, on a boundary surface with outward normal $\mathbf{n}$ may be decomposed into the normal component, $w_n = \mathbf{w} \cdot \mathbf{n}$, and tangential component, $\mathbf{w}_\tau = (\mathbf{I} - \mathbf{n} \otimes \mathbf{n}) \cdot \mathbf{w} = \mathbf{w} - w_n \mathbf{n}$. The surface load *fluid normal velocity* may be used to prescribe a desired value of w_n on a boundary surface.

```
<surface_load type="fluid normal velocity" surface="surface1">
  <value lc="1">1.0</value>
  <velocity>1</velocity>
</surface_load>
```

Optionally, FEBio can generate a parabolic velocity profile over the named surface, This option can be turned on using the *parabolic* element (0=default, 1=parabolic). In this case, the *velocity* element provides the average value of this parabolic velocity profile, while *value* remains a scale factor optionally tied to a loadcurve. The parabolic profile is calculated under the assumption of steady Poiseuille flow in an infinite tube whose cross-section has the shape of the named surface. The parabolic velocity distribution reduces to zero only on portions of the boundary curve of the named surface where the user has fixed all three components of the fluid velocity (no-slip condition). On remaining portions of this boundary curve, symmetry conditions are assumed for the three-dimensional parabolic normal velocity profile.

```
<surface_load type="fluid normal velocity" surface="surface1">
  <velocity lc="1">-1.0</velocity>
  <parabolic>1</parabolic>
</surface_load>
```

If an arbitrary distribution with different scalar value needs to be specified for each facet, use the following syntax,

```
<surface_load type="fluid normal velocity" surface="surface1">
  <value surface_data="inlet"/>
  <velocity lc="1">-1.0</value>
</surface_load>
```

and provide the facet values in a user-defined *SurfaceData* map with *name="inlet"* and *data_type="scalar"* (Section 3.8.1).

By default, this condition only prescribes the natural boundary condition w_n, leaving $\mathbf{w}_\tau$ free. Optionally, it is possible to also enforce $\mathbf{w}_\tau = \mathbf{0}$ by prescribing essential boundary conditions on the wx, wy and wz components of $\mathbf{w} = w_n\mathbf{n}$ at the nodes of each facet, based on the value of $\mathbf{n}$ at each node (obtained by averaging $\mathbf{n}$ from adjoining facets). This option may be turned on using the *prescribe_nodal_velocities* element (0=default, 1=prescribed), for example,

```
<surface_load type="fluid normal velocity" surface="surface1">
  <velocity>-1.0</velocity>
  <prescribe_nodal_velocities>1</prescribe_nodal_velocities>
</surface_load>
```

When the fluid is prescribed on a surface boundary, better numerical convergence is achieved if the fluid dilatation is also prescribed on the rim (the boundary curve) of that surface boundary. While this can be done explicitly, a better option is to set the fluid dilatation on the rim to match the average pressure on the surface boundary, calculated during the analysis. To select this option, use the *prescribe_rim_pressure* element (0=default, 1=prescribed), for example,

```
<surface_load type="fluid normal velocity" surface="surface1">
  <velocity>-1.0</velocity>
  <parabolic>1</parabolic>
  <prescribe_nodal_velocities>1</prescribe_nodal_velocities>
  <prescribe_rim_pressure>1</prescribe_rim_pressure>
</surface_load>
```

Note that this *prescribe_rim_pressure* option will not work if the mesh of the boundary surface has no internal nodes.

3.13.2.18 Fluid Velocity

In a fluid or fluid-FSI analysis, this boundary condition prescribes the fluid velocity vector relative to the mesh, $\mathbf{w}$, on a named surface, simultaneously satisfying the natural boundary condition for $w_n = \mathbf{w} \cdot \mathbf{n}$ and prescribing essential boundary conditions on the wx, wy and wz components of $\mathbf{w}$ at the nodes of each facet, based on the value of $\mathbf{n}$ at each node (obtained by averaging $\mathbf{n}$ from adjoining facets).

```
<surface_load type="fluid velocity" surface="surface1">
  <scale lc="1">1.0</scale>
  <velocity>0,0,1</velocity>
</surface_load>
```

If a different vector value needs to be specified for each facet, use the following syntax,

```
<surface_load type="fluid velocity" surface="surface1">
  <scale lc="1">1.0</scale>
  <velocity surface_data="inlet"/>
</surface_load>
```

and provide the facet values in a user-defined *SurfaceData* map with *name="inlet"* and *data_type="vec3"* (Section 3.8.1).

3.13.2.19 Fluid Rotational Velocity

This boundary condition prescribes a rotational velocity on an axisymmetric surface. It is the user's responsibility to ensure that the surface is axisymmetric. The angular velocity $\boldsymbol{\omega} = \omega\mathbf{n}$ is prescribed by specifying the angular speed ω and the unit vector along the axis of rotation (axis of symmetry) $\mathbf{n}$. The user-defined vector $\mathbf{n}$ is automatically normalized. If the axis does not pass through the origin, the user may specify any point $\mathbf{p}$ on the axis. The fluid velocity relative to the mesh, $\mathbf{w}$, for nodes on the selected surface is evaluated from $\mathbf{w} = \boldsymbol{\omega} \times \mathbf{r}$, where $\mathbf{r} = (\mathbf{I} - \mathbf{n} \otimes \mathbf{n}) \cdot (\mathbf{x} - \mathbf{p})$ is the shortest vector from the axis to the node and $\mathbf{x}$ is the nodal position.

```
<surface_load type="fluid rotational velocity" surface="FluidRotationalVelocity01">
<angular_speed lc="1">1</angular_speed>
<axis>0,0,1</axis>
<origin>0,0,0</origin>
</surface_load>
```

Though this boundary condition may be used in fluid-FSI analyses, it is most relevant for standard fluid analyses where the mesh is stationary.

3.13.2.20 Fluid-FSI Traction

In a fluid-FSI analysis, the fluid traction across an interface between a *fluid-FSI* material and a solid domain must be prescribed explicitly as a traction boundary condition on the solid domain. Otherwise, the solid domain cannot sense the fluid pressure and viscous traction. Users must identify each surface on which this FSI surface load must be prescribed. There are no user-defined parameters for this surface load.

```
<surface_load type="fluid-FSI traction" surface="surface1">
<surface_load>
```

This surface load must also be prescribed on free fluid surfaces (such as the surface of the fluid in open channel flow) to allow those surfaces to deform in response to the fluid stresses (e.g., producing surface waves). (Do not apply this surface load on inlet or outlet boundaries through which fluid flows.)

3.13.2.21 Biphasic-FSI Traction

In a fluid-FSI analysis, the fluid traction across an interface between a *biphasic-FSI* material and a solid domain must be prescribed explicitly as a traction boundary condition on the solid domain. Otherwise, the solid domain cannot sense the fluid pressure and viscous traction. Users must identify each surface on which this FSI surface load must be prescribed. There are no user-defined parameters for this surface load.

```
<surface_load type="biphasic-FSI traction" surface="surface1">
<surface_load>
```

This surface load must also be prescribed on free fluid surfaces (such as the surface of the fluid in open channel flow) to allow those surfaces to deform in response to the fluid stresses (e.g., producing surface waves). (Do not apply this surface load on inlet or outlet boundaries through which fluid flows.)

3.13.2.22 Solute Backflow Stabilization

In a fluid-solutes analysis this surface load checks for the presence of solvent (fluid) backflow, i.e., when $v_n^f = \mathbf{v}^f \cdot \mathbf{n}$ is negative. When this condition is encountered, the effective concentration of the selected solute (*solute_id*) is set to its converged value from the previous time step. Otherwise, the natural boundary condition $\tilde{j}_n = 0$ (zero diffusive solute flux relative to the solvent) prevails.

```
<surface_load type="solute backflow stabilization" surface="surface1">
<solute_id>neutral</solute_id>
</surface_load>
```

3.13.3 Body Loads

A body load can be used to define a source term to the model. The following sections define the different type of body loads that can be applied. In the case of body forces, these body loads represent the body force per mass, $\mathbf{b}$, which appears in the momentum balance as

$$\rho\mathbf{a} = \operatorname{div}\boldsymbol{\sigma} + \rho\mathbf{b}\,,$$

where ρ is the mass density, $\mathbf{a}$ is the acceleration (set to zero in quasi-static analyses), and $\boldsymbol{\sigma}$ is the Cauchy stress tensor.

3.13.3.1 Constant Body Force

This constant body force per mass $\mathbf{b}$ is defined as a 3D vector. Each component can be associated with a load curve to define a time dependent body force. Only the non-zero components need to be defined. This type of body force is spatially homogeneous, though it may vary with time when associated with a load curve:

```
<body_load type="const">
  <x lc="1">1.0</x>
  <y lc="2">1.0</y>
  <z lc="3">1.0</z>
</body_load>
```

The *lc* attribute defines the load curve to use for the corresponding component. The values of the components can be used to define scale factors for the load values.

3.13.3.2 Non-Constant Body Force

This body force per mass $\mathbf{b}$ may be spatially inhomogeneous. The spatial inhomogeneity may be specified using a formula with variables X, Y, Z. For example:

```
<body_load type="non-const">
  <x>X+Y+Z</x>
  <y>X-Y-Z</y>
  <z>X*X+Z</z>
</body_load>
```

3.13.3.3 Centrifugal Body Force

A centrifugal body force per mass $\mathbf{b} = \omega^2 r \mathbf{e}_r$ may be used for bodies undergoing steady-state rotation with angular speed ω about a rotation axis directed along $\mathbf{n}$ and passing through the rotation center $\mathbf{c}$. Here, r represents the distance of each material point from the axis of rotation, and $\mathbf{e}_r$ is the radial unit vector from the axis of rotation, both of which are calculated internally from the knowledge of $\mathbf{n}$ and $\mathbf{c}$.

```
<body_load type="centrifugal">
  <angular_speed>1.0</angular_speed>
  <rotation_axis>0.707,0.707,0</rotation_axis>
  <rotation_center>0,0,0</rotation_center>
</body_load>
```

3.13.3.4 Heat Source

A heat source can be defined using the *heat_source* type.

```
<body_load type="heat_source">
  <Q>1.0</Q>
</body_load>
```

3.13.3.5 Surface Attraction

A surface attraction body force can be used to attract a mesh toward a boundary surface, for the purpose of creating a mesh bias. This can be useful for fluid analyses where a finer mesh is needed in the vicinity of a no-slip boundary. The benefit of this tool over other existing tools for creating a biased mesh is its ability to work with structured and unstructured meshes, as well as linear and higher-order elements.

For each element in the mesh, its shortest distance r to the selected boundary surface is evaluated using a surface projection algorithm. Then, an attractive body force per mass,

$$\mathbf{b} = s \exp\left(-\frac{r}{\rho}\right) \mathbf{n}$$

is evaluated along the unit vector $\mathbf{n}$ directed along the shortest projection distance. Here, ρ is a characteristic boundary layer thickness (*blt*) and s is a scale factor for the body force (*bsf*), whose value may be adjusted relative to account for the stiffness of the elastic solid material assigned to the mesh.

```
<body_load type="surface attraction" elem_set="EB1" surface="PressureLoad1">
    <blt>0.5</blt>
    <bsf lc="1">20</bsf>
    <search_radius>0.1</search_radius>
    <search_tol>0.01</search_tol>
</body_load>
```

Note that this body force requires the specification of an attractive surface; optionally, it also accepts the specification of the element set subjected to this attractive body force. The *search_radius* and *search_tol* parameters are used for the projection algorithm. They have the same meaning as in Section 3.14.1.

3.14 Contact Section

The *Contact* section defines all the contact interfaces. Contact boundary conditions are defined with the *contact* sub-element. The *type* attribute specifies the type of contact interface that is defined. The *surface_pair* attribute defines the surface pair to use for this contact definition. The surface pair is defined the the *Mesh* section. For example:

```
<contact type="sliding-elastic" surface_pair="contact1">
  <!-- parameter go here -->
</contact>
```

The *type* can be one of the following options:

Type	Description
sliding-node-on-facet, sliding-facet-on-facet, sliding-elastic, sliding-biphasic, sliding-biphasic-solute, sliding-multiphasic	A sliding interface that may separate (with biphasic contact for *sliding*-biphasic, biphasic-solute contact for *sliding-biphasic-s*olute, and multiphasic contact for *sliding-multiphasic*)
rigid_wall	A sliding interface with rigid wall as secondary surface
rigid_joint	A joint between two rigid bodies
tied,sticky, tied-biphasic	A tied interface (solid-solid, solid-rigid, solid-biphasic, rigid-biphasic) or tied-biphasic interface (biphasic-biphasic).
contact potential	A potential based sliding contact interface

3.14.1 Sliding Interfaces

A sliding interface can be used to setup a non-penetration constraint between two surfaces. As of version 1.2, three different sliding contact algorithms are available. Although all three are based on the same contact enforcement method, they all differ slightly in their implementation and have been shown to give different performance for different contact scenarios. Each sliding contact implementation is identified by a different *type* attribute.

sliding-node-on-facet (N2F) This is FEBio's original implementation of sliding contact. It is based on Laursen's contact formulation [53] which poses the contact problem as a nonlinear constrained optimization problem. In FEBio, the Lagrange multipliers that enforce the contact constraints are computed either using a penalty method or the augmented Lagrangian method.

sliding-facet-on-facet (F2F) This implementation is identical to the *sliding-node-on-facet* implementation but uses a more accurate integration rule: where the former method uses nodal integration, this method uses Gaussian quadrature to integrate the contact equations. This method has been demonstrated to give additional stability and often converges when the former method does not.

sliding-elastic (SE) This sliding contact interface also uses facet-on-facet contact but differs in the linearization of the contact forces, which results in a different contact stiffness matrix compared to the previous two methods. It can be used for frictionless or frictional contact. It may optionally be set to sustain tension to prevent contact surfaces from separating along the direction normal to the interface, while still allowing tangential sliding. This method sometimes performs better than the previous two methods for problems that are dominated by compression. However, the formulation is inherently non-symmetric and therefore will require additional memory and running time. A symmetrized version of this implementation is available (see below), but the symmetric version does not converge as well as the non-symmetric version.

sliding-biphasic (SBP) This method extends the *sliding-elastic* algorithm to support frictionless biphasic contact (see the next section). This method is described in detail in [11] for the frictionless case, and in [**?**] for the frictional algorithm.

sliding-biphasic-solute (SBS) This method is similar to *sliding-biphasic*. This contact implementation supports biphasic-solute contact (see the next section) [14]. When using biphasic-solute materials, the non-symmetric version must be used.

sliding-multiphasic (SMP) This method is similar to *sliding-biphasic-solute*. This contact implementation supports multiphasic contact (see the next section) [14]. When using multiphasic materials, the non-symmetric version must be used.

contact potential (CP) The "contact potential" contact interface implements the method by [47]. This method defines a contact potential that generates a (repulsing) force that prevents penetration from occurring. This contact formulation is not based on the augmented Lagrangian formulation and does not have any of the parameters in the table below. Instead, please see section 3.14.10 for more details on this method.

The following table lists the properties that are defined for sliding interfaces (except contact potential). It is important to note that the three different sliding implementations cannot be used interchangeably: not all features are available for each method. The third, fourth and fifth column indicate if a parameter is available for a particular implementation.

Parameter	Description	N2F	F2F	SE	SBP	SBS SMP	Default
penalty	normal penalty scale factor (1)	•	•	•	•	•	1.0
auto_penalty	auto-penalty calculation flag (2)	•	•	•	•	•	0
update_penalty	updated auto-penalty calculation at each step (2)		•	•	•	•	0
two_pass	two-pass flag (3)	•	•	•	•	•	0
laugon	augmented Lagrangian flag (4)	•	•	•	•	•	0
tolerance	aug. Lagrangian convergence tolerance (4)	•	•	•	•	•	1.0
gaptol	tolerance for gap value (4)	•	•	•	•	•	0.0 (off)

minaug	minimum number of augmentations (4)	•	•	•	•		0
maxaug	maximum number of augmentations (4)	•	•	•	•		10
fric_coeff	frictional coefficient (5)	•		•	•		0.0
fric_penalty	tangential penalty factor (5)	•					0.0
ktmult	tangential stiffness multiplier (5)	•		•			1.0
seg_up	maximum number of segment updates (6)	•		•	•		0 (off)
smooth_aug	smoothed Lagrangian augmentation (7)		•	•	•	•	0 (off)
smooth_fls	smoothed fluid load support for friction calculation (7)				•		0 (off)
contact_frac	solid-on-solid contact area fraction for friction calculation (5)				•		0
symmetric_stiffness	symmetric stiffness matrix flag (8)			•	•	•	0 (off)
search_tol	Projection search tolerance (9)	•	•	•	•	•	0.01
search_radius	search radius (10)		•	•	•	•	1.0
tension	tension flag (11)			•			0
flip_primary	flip normal on primary surface			•	•		0 (off)
flip_secondary	flip normal on secondary surface			•	•		0 (off)
shell_bottom_primary	contact with back of shell on primary surface			•	•		0 (off)
shell_bottom_secondary	contact with back of shell on secondary surface			•	•		0 (off)
offset	offset distance between contacting surfaces (12)			•			

Comments:

1. If the augmented Lagrangian flag is turned off (see comment 4), the penalty method is used to enforce the contact constraint. In this method, the contact traction is determined by the gap (i.e. penetration distance) multiplied by the user-defined *penalty* factor. In the augmented Lagrangian method, the *penalty* parameter is also used but has a slightly different meaning. In this case, it scales the Lagrange multiplier increment. Due to the different meanings, the user might have to adjust the penalty factor when switching between penalty method and augmented Lagrangian method. In general the penalty method requires a larger penalty factor to reach the same gap than the augmented Lagrangian method. See comment 4 for more information on when to use which method.

2. Choosing a good initial penalty parameter can often be a difficult task, since this parameter depends on material properties as well as on mesh dimensions. For this reason, an algorithm has been implemented in FEBio that attempts to calculate a good initial value for the penalty factor $\varepsilon_i$for a particular node/integration piont i on the contact interface:

$$\varepsilon_i = \frac{EA}{V}.$$

Here, A is the area of the element the integration point belongs to, V is the element volume and E is a measure of the elasticity modulus, which is calculated from the elasticity tensor of the element. Although the meaning of E depends on the precise material formulation, in general one can regard it as a measure of the small strain Young's modulus of the material, when *update_penalty* is 0. If update_penalty is set to 1, E is recalculated at each iteration to account for the current deformation.

To use this feature, add the following element to the contact section:

```
<auto_penalty>1</auto_penalty>
```

When the auto-penalty flag is on, the value of the *penalty* parameter serves as a scale factor for the automatically-calculated penalty factor.

3. Each sliding interface consists of a *primary* surface and a *secondary* surface. The primary surface is the surface over which the contact equations are integrated and on which the contact tractions are calculated. The secondar surface is used to measure the gap function and to define the necessary kinematic quantities such as surface normals and tangents. This approach is usually referred to as the *single-pass* method. When using the single-pass algorithm, the results can be influenced by the choice of primary and secondary surfaces. It is best to use the most tessellated surface as the primary and the coarsest as the secondary surface. To resolve the bias issue, one can also use a *two-pass* algorithm for enforcement of the contact constraint. In this case, a single pass is performed first, after which the primary and secondary surfaces are swapped and another pass is performed. When using the two-pass method, the definition of primary and secondary surfaces is arbitrary. In most problems, the single pass is sufficient to enforce contact; with a judicious choice of primary-secondary pair and contact parameters, good results can be obtained. If however, the single pass does not give good answers, for example, when due to the geometry's curvature the gap cannot be small enough with a single pass, the two-pass method can be used, although at the expense of more calculations.

If one of the contacting surfaces is rigid, a slightly different approach is recommended. In this case, it is best to pick the rigid surface as the secondary surface and to use a single pass algorithm. The reason is that the nodal degrees of freedom on the rigid surface are condensed to the rigid degrees of freedom and if the rigid surface is the primary surface, the reaction forces may not propagate correctly to the secondary surface. This is especially true if the rigid degrees of freedom are fixed.

4. In the presence of a sliding interface (and other contact interfaces), FEBio needs to calculate the contact tractions that prevent the two participating surfaces from penetrating. In general these tractions can be found using the method of Lagrange multipliers. However, the direct calculation of these multipliers has several computational disadvantages and therefore FEBio approximates the multipliers using one of two alternative methods: the penalty method and

the augmented Lagrangian method. In the former method, the multipliers are approximated by the gap (i.e. penetration distance) scaled by a suitably chosen penalty factor. In many cases, this method is sufficient to get good results. Since the correctness of a contact solution is directly determined by the amount of penetration at the converged state, the user has direct control over the quality of the solution. By increasing the penalty factor, the penetration is reduced. However, in some cases, especially in large compression problems, the penalty factor required to achieve an acceptable amount of penetration has to be so large that it causes numerical instabilities in the non-linear solution algorithm due to ill-conditioning of the stiffness matrix. In these cases, the augmented Lagrangian method might be a better choice. In this method, the multipliers are determined iteratively where, in each iteration, the multiplier's increments are determined with a penalty-like method. The advantage of this method is twofold: due to the iterative nature, the method will work with a smaller penalty factor, and in the limit, the exact Lagrange multipliers can be recovered.

To turn on the augmented Lagrangian method, simply add the following line to the contact section:

```
<laugon>1</laugon>
```

To turn off the augmented Lagrangian method, either set the value to 0 or remove the parameter altogether. The convergence tolerance is set as follows:

```
<tolerance>0.01</tolerance>
```

With this parameter set, the augmented Lagrangian method will iterate until the relative increment in the multipliers is less than the tolerance. For instance, setting the tolerance parameter to 0.01 implies that the augmented Lagrangian method will converge when there is less than a 1% change in the L2 norm of the augmented Lagrangian multiplier vector between successive augmentations. Alternatively, the user can also specify a tolerance for the gap value. In this case, the iterations will terminate when the gap norm, which is defined as the averaged L2 norm, ($\sqrt{\sum_i < g_i >^2}/N$, $< \cdot >$the Macauley Bracket) is less than the user-specified value:

```
<gaptol>0.001</gaptol>
```

However, one must be careful when specifying a gap tolerance. First note that the gap tolerance is an absolute value (unlike the *tolerance* which is a relative value), so this parameter depends on the dimensions of the model. Also, there are cases when a gap tolerance simply cannot be reached due to the geometry of the model in which case the augmentations may never converge.

Note that both convergence parameters may be defined at once. In that case, FEBio will try to satisfy both convergence criteria. On the other hand, setting a value of zero will turn off the convergence criteria. For example, the default value for *gaptol* is zero, so that FEBio will not check the gap norm by default.

Finally, the *minaug* and *maxaug* can be used to set a minimum and maximum number of augmentations. When the *maxuag* value is reached, FEBio will move on to the next timestep, regardless of whether the force and gap tolerances have been met. When specifying a value for *minaug*, FEBio will perform at least minaug augmentations.

5. The *sliding-node-on-facet, sliding-elastic*, and *sliding-biphasic* contact implementations support friction. For *node-on-facet*, three parameters control the frictional response: *fric_coeff* is the material's friction coefficient and its value must be in the range from 0.0 to 1.0; *fric_penalty* is the penalty factor that regulates the tangential traction forces, much like the *penalty* parameter regulates the normal traction force component; the parameter *ktmult* is a scale factor for the tangential stiffness matrix. It is default to 1.0, but it is observed that reducing this value might sometimes improve convergence. For *sliding-elastic* and *sliding-biphasic* only *fric_coeff* is needed to use frictional contact. In addition, for *sliding-biphasic*, the friction algorithm requires the specification of the solid-on-solid contact area fraction, which is a number between 0 and 1 (see *FEBio Theory Manual*). Frictional contact is inherently dissipative and thus history-dependent. The solution may vary with the size of time increments, the cycle of loading, or Lagrangian augmentations (when *laugon*=1).

6. In a contact problem, FEBio calculates the projection of each node of the primary surface onto the secondary surface. As a node slides across the secondary surface, the corresponding surface facet can change. However, in some cases, switching facets is undesirable since it might cause instabilities in the solution process or a state in which the node oscillates continuously between two adjacent facets and thus prevents FEBio from meeting the displacement convergence tolerance. The parameter *seg_up* allows the user to control the number of segment updates FEBio will perform during each time step. For example, a value of 4 tells FEBio it can do the segment updates during the first four iterations. After that, nodes will not be allowed to switch to new segments. The default value is 0, which means that FEBio will do a segment update each iteration of each timestep.

7. The augmented Lagrangian method may produce a non-smooth contact pressure distribution at the contact surface, even though stresses within the underlying elements may remain relatively smoother. Turning this flag on changes the method of updating the Lagrange multiplier to use the projection of the element stress to the corresponding contact face. This feature only works with some element types (HEX8G8, HEX8G1, TET4G4, TET4G1, PENTA6G6, TET10G1, TET10G4, TET10G8, TET10GL11, TET15G4, TET15G8, TET15G11, TET15G15, PYRA5G8, PYRA13G8).

   In the *sliding-biphasic* friction algorithm the friction coefficient depends on the fluid pressure and load support. Smoothing this fluid load support may produce a smoother distribution of the friction coefficient over the contact interface.

8. The *sliding-elastic, sliding-biphasic*, *sliding-biphasic-solute* and *sliding-multiphasic* contact implementations for sliding contact are inherently non-symmetric formulations. Symmetrized versions of these algorithms do not perform as well as the nonsymmetric version so it is recommended to use the latter. The following line in the *contact* element controls which version of the algorithm is used.

   ```
   <symmetric_stiffness>0</symmetric_stiffness>
   ```

   A value of 1 uses the symmetric version, whereas a value of 0 uses the non-symmetric version. A similar line may be included in the *Control* element to use a non-symmetric global stiffness matrix. In some cases, a non-symmetric contact stiffness may converge well even when the global stiffness matrix is symmetric.

9. The *search_tol* parameter defines the search tolerance of the algorithm that projects nodes of the primary surface onto facets of the secondary syrface. A node that falls outside an

element, but whose distance to the closest element's edge is less than the search tolerance is still considered inside. This can alleviate convergence problems when nodes are projected onto edges of elements and due to numerical error may be projected outside the surface.

10. The *search_radius* is a dimensional search radius used by the algorithm that projects points onto facets. When the distance between the point and the facet exceeds the dimensional search radius, that projection is ignored. This can alleviate convergence problems when surfaces have multiple folds and the projection produces multiple solutions, only one of which (the closest distance) is valid.

11. The *tension* flag determines whether the contact interface can sustain tension and compression (*tension*=1) or only compression (*tension*=0).

12. The *offset* distance makes it possible to maintain the contact surfaces at a distance. This option is useful, for example, in fluid-structure interaction problems, where solid structures separated by a fluid domain are pushed against each other. Specifying a contact interface between the (internal) surfaces of the solid structures would prevent them from overlapping, but adding an offset distance will also prevent the fluid mesh from getting squished too thin or from inverting. These internal surfaces would also typically have FSI Traction interfaces defined on them (see Sections 3.13.2.20-3.13.2.21).

3.14.2 Biphasic Contact

The *sliding-biphasic* implementation for sliding interfaces can deal with biphasic contact surfaces (including biphasic-on-biphasic, biphasic-on-elastic, biphasic-on-rigid) [11**?**]. It allows for the possibility to track fluid flow across the contact interface. In other words, fluid can flow from one side of the contact interface to the other when both contact surfaces are biphasic. To use this feature, the user must define an additional contact parameter, namely:

```
<pressure_penalty>1.0</pressure_penalty>
```

In the same way that the *penalty* parameter controls the contact tractions, this parameter controls the penalty value that is used to calculate the Lagrange multipliers for the pressure constraint. If the *laugon* flag is set, the augmented Lagrangian method is used to enforce the pressure constraint. And if the *auto_penalty* flag is defined (which is the recommended approach), an initial guess for the pressure penalty is calculated automatically using the following formula:

$$\varepsilon_p = \frac{kA}{V},$$

where A is the element's area, V is the element's volume and k is a measure of the permeability which is defined as one third of the trace of the material's initial permeability tensor.

When either contact surface is biphasic, the surface outside the contact area(s) is automatically set to free-draining conditions (equivalent to setting the fluid pressure to zero).

As detailed in [**?**] and the *FEBio Theory Manual*, the *sliding-biphasic* contact algorithm can include frictional contact. The effective friction coefficient μ_{eff} in this frictional contact algorithm depends on the temporally evolving local fluid load support (the ratio of fluid pressure p to the normal component of the mixture contact traction t_n), according to

$$\mu_{\text{eff}} = \mu_{\text{eq}} \left(1 + (1-\varphi)\,\frac{p}{t_n}\right) .$$

Recall that t_n is negative whereas p is positive in compression. Here, μ_{eq} (*fric_coeff*) is the friction coefficient in the limit when the fluid load support has reduced to zero, and φ (*contact_frac*) is the fraction of the apparent contact area over which the solid matrix of the primary surface contacts that of the secondary surface. Thus, $1-\varphi$ is the fraction of the apparent contact area where fluid contacts fluid or solid. For perfectly smooth contact surfaces one may assume that $\varphi = \varphi_1^s \varphi_2^s$, where φ_1^s and φ_2^s are the solid area (or volume) fractions of the primary and secondary contact surfaces, respectively. For example, when both contact surfaces are non-porous ($\varphi_1^s = \varphi_2^s = 1$), the effective friction coefficient reduces to $\mu_{\text{eff}} = \mu_{\text{eq}}$.

When performing biphasic-on-rigid contact a two-pass analysis should not be used; the rigid surface should be the secondary surface.

3.14.3 Biphasic-Solute and Multiphasic Contact

The *sliding-biphasic-solute* implementation for sliding interfaces can deal with biphasic-solute contact surfaces (including biphasic-solute-on-biphasic-solute, biphasic-solute-on-biphasic, biphasic-solute-on-elastic, biphasic-solute-on-rigid) and the *sliding-multiphasic* contact interface can similarly deal with multiphasic contact surfaces. These contact interfaces allow for the possibility to track fluid and solute flow across the contact interface [14]. In other words, fluid and solute can flow from one side of the contact interface to the other. To use this feature, the user must define additional contact parameters, namely:

```
<pressure_penalty>1.0</pressure_penalty>
<concentration_penalty>1.0</concentration_penalty>
<ambient_pressure>0</ambient_pressure>
<ambient_concentration>0</ambient_concentration> (for sliding-biphasic-solute)
```

or

```
<ambient_concentration sol="id">0</ambient_concentration> (for sliding-multiphasic)
```

In the same way that the *penalty* parameter controls the contact tractions, these penalty parameters control the penalty values that are used to calculate the Lagrange multipliers for the pressure and concentration constraints. If the *laugon* flag is set, the augmented Lagrangian method is used to enforce the pressure and concentration constraints. And if the *auto_penalty* flag is defined, an initial guess for the pressure and concentration penalty is calculated automatically using the following formulas:

$$\varepsilon_p = \frac{k \cdot A}{V}, \quad \varepsilon_c = \frac{d \cdot A}{V},$$

where A is the element's area, V is the element's volume, k is a measure of the fluid permeability which is defined as one third of the trace of the material's initial permeability tensor, and d is a measure of the solute diffusivity which is defined as one third of the trace of the material's initial diffusivity tensor.

When either contact surface is biphasic-solute or multiphasic, the surface outside the contact area(s) is automatically set to ambient conditions (equivalent to setting the effective fluid pressure and effective solute concentration to the <ambient_pressure> and <ambient_concentration> values, respectively). Ambient conditions may also be associated with a load curve, for example:

```
<ambient_pressure lc="2">1.0</ambient_pressure>
<ambient_concentration lc="3">1.0</ambient_concentration>
```

When performing biphasic-solute-on-rigid or multiphasic-on-rigid contact, a two-pass analysis should not be used; the rigid surface should be the secondary surface.

3.14.4 Rigid Wall Interfaces

A rigid wall interface is similar to a sliding interface, except that the secondary surface is a rigid wall. The following properties are defined for rigid wall interfaces:

Parameter	Description	Default
laugon	Augmented Lagrangian flag	0 (false)
tolerance	augmentation tolerance	0.01
penalty	penalty factor	1.0
plane	the plane equation for the rigid wall	N/A
offset	the normal offset of the plane defined by the *plane* parameter	0.0

The *plane* property defines the reference plane for the rigid wall. Its value is an array of four values:a, b, c, d_0. The actual plane is defined by specifying the *offset* to the reference plane. The offset parameter takes a loadcurve as an optional attribute to define the motion of the plane as a function of time. The loadcurve defines the offset h from the initial position in the direction of the plane normal:

$$ax + by + cz + d\,(t) = 0, \quad d\,(t) = d_0 + h\,(t)$$

So, for example, a rigid wall that initially lies in the xy-coordinate plane and moves in the z-direction would be specified as follows:

```
<plane>0,0,1,0</plane>
<offset lc="1">1.0</offset>
```

3.14.5 Tied Interfaces

A tied interface can be used to connect two non-conforming meshes [54]. A tied interface requires the definition of both a primary and a secondary surface. It is assumed that the nodes of the primary surface are connected to the faces of the secondary surface. The following control parameters need to be defined:

<penalty>	penalty factor
<tolerance>	augmentation tolerance

3.14.6 Tied Elastic Interfaces

A tied elastic interface is similar to the tied interface. It may be used for tying solid materials. It enforces continuity of the displacement across the interface. The following control parameters need to be defined:

Parameter	Description	Default
penalty	normal penalty scale factor (1)	1.0
pressure_penalty	pressure penalty scale factor	1.0
auto_penalty	auto-penalty calculation flag (2)	0
update_penalty	update auto-penalty calculation flag (2)	0
two_pass	two-pass flag (3)	0
laugon	augmented Lagrangian flag (4)	0
tolerance	aug. Lagrangian convergence tolerance (4)	1.0
gaptol	tolerance for gap value (4)	0.0 (off)
symmetric_stiffness	symmetric stiffness matrix flag (7)	0
search_tol	Projection search tolerance (8)	0.01
search_radius	search radius (10)	1.0

3.14.7 Tied Biphasic Interfaces

A tied biphasic interface is similar to the tied interface. It may be used for tying any combination of solid, biphasic, and rigid materials. It enforces continuity of the fluid pressure across the interface when both materials are biphasic. The following control parameters need to be defined:

Parameter	Description	Default
penalty	normal penalty scale factor (1)	1.0
pressure_penalty	pressure penalty scale factor	1.0
auto_penalty	auto-penalty calculation flag (2)	0
update_penalty	update auto-penalty calculation flag (2)	0
two_pass	two-pass flag (3)	0
laugon	augmented Lagrangian flag (4)	0
tolerance	aug. Lagrangian convergence tolerance (4)	1.0
gaptol	tolerance for gap value (4)	0.0 (off)
symmetric_stiffness	symmetric stiffness matrix flag (7)	0
search_tol	Projection search tolerance (8)	0.01
search_radius	search radius (10)	1.0

3.14.8 Tied Multiphasic Interfaces

A tied multiphasic interface is similar to the tied biphasic interface. It may be used for tying any combination of solid, biphasic, multiphasic and rigid materials. It enforces continuity of the effective fluid pressure and effective solute concentrations across the interface when both materials are biphasic or multiphasic. The following control parameters need to be defined:

Parameter	Description	Default
penalty	normal penalty scale factor (1)	1.0
pressure_penalty	pressure penalty scale factor	1.0
concentration_penalty	concentration penalty scale factor	1.0
auto_penalty	auto-penalty calculation flag (2)	0
update_penalty	update auto-penalty calculation flag (2)	0
two_pass	two-pass flag (3)	0
laugon	augmented Lagrangian flag (4)	0
tolerance	aug. Lagrangian convergence tolerance (4)	1.0
gaptol	tolerance for gap value (4)	0.0 (off)
symmetric_stiffness	symmetric stiffness matrix flag (7)	0
search_tol	Projection search tolerance (8)	0.01
search_radius	search radius (10)	1.0

3.14.9 Sticky Interfaces

A sticky interface is similar to a tied interface except that it allows for initial separation of the tied surfaces and breaking of the tie after a user-defined normal traction is exceeded. The tie is only applied when the surfaces contact and sustained as long as the normal traction is less than the threshold.

Parameter	Description
laugon	augmentation flag
penalty	penalty factor
tolerance	augmentation tolerance
minaug	minimum number of required augmentations
maxaug	maximum number of augmentations
search_tolerance	tolerance for nodal projection onto secondary surface facet
max_traction	threshold for normal traction (1)
snap_tol	minimum distance for tie activation (2)

The contact surfaces are defined as in the *tied* interface (see Section 3.14.5).
Comments:

1. The *max_traction* parameter can be used to break the tied interface after the normal traction exceeds the specified value. Initially, this value is set to zero, in which case FEBio will ignore this value and the tie cannot be broken.

2. The *snap_tol* parameter is used in determining the minimum distance that a primary surface node must have approached the secondary surface facet in order to snap onto the secondary surface. The initial value is zero, meaning a node must have penetrated the secondary surface before it will be tied to it.

3.14.10 Contact Potential

The contact potential method is based on the formulation by [47]. In this formulation, a repulsive potential is defined between two opposing surfaces. A force is generated that aims to prevent

physical contact between the two surfaces. As a result, there will always be a small gap between the contacting surfaces. Note that this is the opposite of what happens for the other sliding contact interfaces, which require physical contact (and even a little bit of penetration) before a contact force can be generated.

The type attribute for this sliding interface is "*contact potential*" amd the following parameters can be defined.

Parameter	Description	Default
kc	force scale factor (1)	0.0
p	order of polynomial that defines the potential (2)	4
R_in	Inner radius (3)	1.0
R_out	Outer radius (3)	2.0
R0_min	Minimum separation in reference frame	0
w_tol	threshold for angle criterion	0

Comments:

1. The *kc* parameter defines the scale factor for the force. This has a similar purpose as the penalty factor in other sliding contact interfaces.
2. The parameter *p* defines the order of the potential. The paper discusses why the value 4 is in a sense an optimal value so it is recommend to use the default value.
3. The potential is divided in an "inner" and "outer" region. If *r* is the distance from a point on the primary contact surface to the opposing secondary surface, then the "inner" region is defined by the criterion $0 < r < R_{in}$and the "outer" region is defined by $R_{in} < r < R_{out}$. The contact force is stronger in the inner region.
4. The *R0_min* parameter defines the minimum distance between two potential contacting points in the reference configuration. In other words, two points whose initial distance was smaller than *R0_min* will not be considered as a valid contact pair. This parameter is important in self-contact to avoid neighboring points from generating contact forces. The default value of 0 indicates that this criterion is not used.
5. The *w_tol* parameter defines a local normal criterion for a potential pair: If the absolute value of the cosine of the angle between the two local normals is **less** than w_tol, the two points are **not** considered in contact. In other words, the two normals have to point in the "same direction" in order for the two contacting points to be in contact. This criterion can be useful in cases of self-contact in order to avoid invalid contact pairs. A value of 0 effectively ignores this criterion.

Example:

```
<contact type="contact potential" name="FacetOnFacetSliding1" surface_pair="FacetOnFacetSli
    <kc>1e-6</kc>
    <p>4</p>
    <R_in>0.01</R_in>
    <R_out>0.05</R_out>
</contact>
```

3.15 Constraints Section

The Constraints section allows the user to enforce additional constraints to the model.

3.15.1 Symmetry Plane

A symmetry plane enforces zero solid displacement normal to a user-selected plane. This constraint is prescribed on a deformable surface of the model and is enforced for every node on that surface. It is the user's responsibility to select only planar surfaces.

```
<constraint type="symmetry plane" surface="SymmetryPlane01">
<laugon>1</laugon>
<penalty>1e6</penalty>
<tol>1e-6</tol>
<minaug>0</minaug>
<maxaug>50</maxaug>
</constraint>
```

The *surface* (SymmetryPlane01 in this example), is defined in the *Mesh* section. Let $\mathbf{u}$ denote the nodal displacement vector and let $\mathbf{n}$ denote the unit normal to the plane; the symmetry plane constraint enforces $u_n \equiv \mathbf{u}\cdot\mathbf{n} = 0$ using a penalty method, optionally with augmented Lagrangian. The reaction force needed to enforce this constraint is evaluated as εu_n, where ε is the user-specified *penalty* parameter (with units of force per length). To use the augmented Lagrangian method, set *laugon* to 1, and the Lagrange multiplier λ will be augmented as $\lambda \leftarrow \lambda + \varepsilon u_n$. Augmentations terminate when the relative change in λ is less than the user-specified tolerance *tol*, or when the number of augmentations exceeds *maxaug*.

3.15.2 Frictionless Fluid Wall

One of the natural boundary conditions for fluid analyses is to enforce zero fluid velocity normal to a boundary, implying that this boundary is a frictionless fluid wall. In some analyses however, this natural boundary condition may not be enforced sufficiently well. In such cases, use this frictionless fluid wall linear constraint to produce a more strict enforcement of this condition.

```
<constraint type="frictionless fluid wall" name="FrictionlessFluidWall1" surface="Frictionl
<laugon>0</laugon>
<tol>0.2</tol>
<penalty>1</penalty>
<minaug>0</minaug>
<maxaug>10</maxaug>
</constraint>
```

In fluid-structure interaction analyses, this constraint may be used in conjunction with the *symmetry plane* described in Section 3.15.1, using a lower value of the *penalty* parameter, as may be necessary in practice.

3.15.3 Fixed Normal Displacement

A fixed normal displacement constraint enforces zero solid displacement normal to a user-selected surface. This constraint is prescribed on a deformable surface of the model and is enforced for every node on that surface. It uses the local surface normal in the selected surface's reference configuration. If the selected surface is planar, the functionality of this constraint is equivalent to the Symmetry Plane constraint described in Section 3.15.1.

```
<constraint type="fixed normal displacement" surface="FixedNormalDisplacement01">
<laugon>0</laugon>
<penalty>1000</penalty>
<tol>0.2</tol>
<minaug>0</minaug>
<maxaug>10</maxaug>
<shell_bottom>0</shell_bottom>
</constraint>
```

The *surface* (FixedNormalDisplacement01 in this example), is defined in the *Mesh* section. Let $\mathbf{u}$ denote the nodal displacement vector and let $\mathbf{n}$ denote the unit normal at each node of the surface; the fixed normal displacement constraint enforces $u_n \equiv \mathbf{u} \cdot \mathbf{n} = 0$ using a penalty method, optionally with augmented Lagrangian. The reaction force needed to enforce this constraint is evaluated as εu_n, where ε is the user-specified *penalty* parameter (with units of force per length). To use the augmented Lagrangian method, set *laugon* to 1, and the Lagrange multiplier λ will be augmented as $\lambda \leftarrow \lambda + \varepsilon u_n$. Augmentations terminate when the relative change in λ is less than the user-specified tolerance *tol*, or when the number of augmentations exceeds *maxaug*.

For example, this constraint may be used on a cylindrical shaft, over the portion of the shaft which is supported by a bearing journal.

3.15.4 Normal Fluid Velocity Constraint

A normal fluid velocity constraint forces the fluid velocity to remain normal to the selected surface. It is enforced for every node on that surface.

```
<constraint type="normal fluid velocity" surface="NormalFlowSurface01">
<laugon>1</laugon>
<penalty>1e6</penalty>
<tol>1e-6</tol>
<minaug>0</minaug>
<maxaug>50</maxaug>
</constraint>
```

The *surface* (NormalFlowSurface01 in this example), is defined in the *Mesh* section. Let $\mathbf{v}$ denote the nodal fluid velocity vector and let $\mathbf{n}$ denote the unit normal at each node; the normal flow constraint enforces $\mathbf{v} = (\mathbf{v} \cdot \mathbf{n})\,\mathbf{n}$, or equivalently, $\mathbf{v}_\tau = (\mathbf{I} - \mathbf{n} \otimes \mathbf{n}) \cdot \mathbf{v} = \mathbf{0}$, using a penalty method, optionally with augmented Lagrangian. The reaction force needed to enforce this constraint is evaluated as $\varepsilon \mathbf{v}_\tau$, where ε is the user-specified *penalty* parameter (with units of force per velocity). To use the augmented Lagrangian method, set *laugon* to 1, and the vectorial Lagrange multiplier

$\boldsymbol{\lambda}$ will be augmented as $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \varepsilon \mathbf{v}_\tau$. Augmentations terminate when the relative change in $\boldsymbol{\lambda}$ is less than the user-specified tolerance *tol*, or when the number of augmentations exceeds *maxaug*.

This constraint should only be used on a surface where the fluid pressure or dilatation has been fixed or prescribed. It is not compatible with boundary conditions that prescribe the fluid velocity. To prescribe a fluid velocity that remains normal to the selected surface, use the surface load *fluid normal velocity* (Section 3.13.2.17).

3.15.5 Uniform Fluid Velocity Constraint

When an inlet boundary is subjected to a prescribed fluid pressure, the profile of the inlet fluid velocity remains unconstrained, and this lack of constraint often leads to poor numerical performance due to the production of velocity spikes. To avoid this problem, use this uniform fluid velocity constraint to help stabilize inlet conditions along the direction normal to the bondary. This constraint evaluates the average fluid velocity normal to the selected boundary, $\bar{v}_n$, then prescribes a linear constraint at every node such that $\mathbf{v} \cdot \mathbf{n} = \bar{v}_n$ is enforced. For example,

```
<constraint name="UniformFluidFlow1" surface="UniformFluidFlow1" type="uniform fluid flow">
<laugon>0</laugon>
<tol>0.1</tol>
<penalty>10000</penalty>
<minaug>0</minaug>
<maxaug>50</maxaug>
</constraint>
```

If a very strong enforcement of this constraint is desired, increase the *penalty* and/or turn Lagrange augmentation on (*laugon*=1). This constraint is not compatible with boundary conditions that prescribe the normal component of the fluid velocity on that same boundary.

3.15.6 The Prestrain Update Rules

The prestrain update rules for are implemented via non-linear constraints in FEBio. This is done because non-linear constraints automatically participate in FEBio's augmentation mechanism, which allows the user to update and rerun the time step. The following section details how to setup the update rules in the FEBio input file and discusses the currently supported rules.

3.15.6.1 Using Update rules

In order to apply an update rule, a constraint definition must be added to the Constraints section of the input file. The type attribute is used to specify which update rule to use.

```
<Constraints>
  <constraint type="[name of update rule]">
    <!-- parameters go here -->
  </constraint>
</Constraints>
```

The following table lists the available update rules.

type	description
prestrain	Eliminate distortion due to incompatibility
in-situ stretch	Enforce the given in-situ fiber stretches

Below, the supported update rules are presented in more detail. All of them share parameters for controlling the convergence of the update algorithm and these shared parameters are listed in the following table.

parameter	description	initial value
update	update flag (1)	1
tolerance	convergence tolerance	0.0
min_iters	minimum number of iterations	0
max_iters	maximum number of iterations	-1 (i.e. ignored)

Comments:

1. By specifying a loadcurve for the update flag, the update can be delayed. This can be useful if, for instance, the prestrain is applied incrementally and the update rule should not be applied until the full prestrain field is applied. In that case, specifying a loadcurve for the update flag that is zero while the prestrain is applied, will delay the update process.

3.15.6.2 prestrain update rule

The idea behind this rule is to eliminate the distortion induced by the incompatibility of the initial prestrain gradient with the reference geometry. Thus, we retain the original reference geometry at the cost of an altered effective prestain field. The update rule is given by the following equation.

$$\mathbf{G}^{k+1} = \mathbf{G}^k \cdot \mathbf{F}_c \tag{3.15.1}$$

This update rule does not define any additional parameters aside the ones from above.
Example:

```
<constraint type="prestrain">
  <tolerance>0.01</tolerance>
</constraint>
```

3.15.6.3 The in-situ stretch update rule

The idea behind this update rule is to enforce the fiber stretch induced by the initial prestrain gradient. As with the in-situ stretch generator option, this rule has an isochoric version and an uniaxial version for the update rule.

$$\mathbf{G}_{iso} = \mathbf{Q}\begin{bmatrix} \lambda^{-1} & & \\ & \lambda^{1/2} & \\ & & \lambda^{1/2} \end{bmatrix}\mathbf{Q}^T, \qquad \mathbf{G}_{uni} = \mathbf{Q}\begin{bmatrix} \lambda^{-1} & & \\ & 1 & \\ & & 1 \end{bmatrix}\mathbf{Q}^T \tag{3.15.2}$$

where $\lambda^2 = \mathbf{a}_0 \cdot \mathbf{C}^k \cdot \mathbf{a}_0$, is the fiber stretch induced by the distortion.

parameter	description	initial value
isochoric	Choose the isochoric update rule or not	1(=true)

Example:

```
<constraint type="in-situ stretch">
  <tolerance>0.01</tolerance>
  <isochoric>1</isochoric>
</constraint>
```

3.16 Discrete Section

This section defines the materials used by the discrete elements and assigns these materials to the discrete elements sets defined in the *Mesh* section. The materials are defined via the *discrete_material* element and the materials are assigned to discrete element sets using the *discrete* element.

```
<Discrete>
  <discrete_material id="1" type="nonlinear spring">
    <force lc="1">1.0</force>
  </discrete_material>
  <discrete_material id="2" type="linear spring">
    <E>2.0</E>
  </discrete_material>
  <discrete dmat="1" discrete_set="springs1"/>
  <discrete dmat="2" discrete_set="springs2">
</Discrete>
```

The *discrete_material* and the *discrete* elemens are defined in more detail below.

3.16.1 Discrete Materials

The *discrete_material* section defines a material that can be assigned to a discrete element set. The *id* attribute defines the material ID and the *type* attribute defines the material type. The following types are currently supported.

Type	Description
linear spring	spring that has a linear force-displacement relation
nonlinear spring	user defines the force-displacement relation
Hill	discrete material that defines a Hill-type muscle model

3.16.1.1 Linear Spring

The linear spring requires the E parameter that defines the spring constant.

3.16.1.2 Nonlinear spring

The nonlinear spring requires the *force* parameter, which defines the loadcurve that will be used for the force-displacement relation.

3.16.1.3 Hill

The Hill discrete material defines the following parameters.

- *Vmax*: maximum shortening velocity
- *ac*: activation level
- *Fmax*: maximum force

- *Ksh*: shape parameter
- *Lmax*: strain (i.e. stretch - 1) at which Fmax occurs.
- *L0*: initial reference length
- *Sv*: max velocity scale
- *Ftl*: normalized tension-length curve
- *Ftv*: normalized tension-velocity curve

The force in the Hill discrete element is the sum of the passive element and the active element.

$$F = F_p + F_a$$

The passive force is given by,

$$F_p = \begin{cases} Fmax\left(\exp\left(Ksh\left(l-1\right)/Lmax-1\right)/\exp\left(Ksh-1\right)\right) & ,l>1 \\ 0 & ,l\leqslant 1 \end{cases}$$

where l is the relative stretch defined by, $l = lm/l0$ and $lm$is the discrete element length and $l0$is either the L0 parameter, or the initial discrete element length (if L0 is set to zero).

The active force is given by,

$$Fa = ac * Fmax * Ftl\left(l\right) * Ftv\left(v\right)$$

Here, v, a measure of the relative discrete element's growth speed, is defined by,

$$v = vm/\left(Vmax * Sv\left(ac\right)\right)$$

where $vm$is the actual discrete element's growth speed.

The properties Sv, Ftl, and Ftv, are optional and will evaluate to 1 if omitted. They can be defined as load curves. See the exampe below.

Example:

```
<discrete_material id="1" name="test" type="Hill">
  <Vmax>1</Vmax>
  <ac>0.1</ac>
  <Fmax>50</Fmax>
  <Ksh>5</Ksh>
  <Lmax>1.5</Lmax>
  <L0>10</L0>
  <Ftl type="point">
    <interpolate>smooth</interpolate>
    <points>
      <pt>0.0, 0</pt>
      <pt>0.1, 0.000258139</pt>
   <pt>0.2, 0.00161616</pt>
   <pt>0.3, 0.00740118</pt>
   <pt>0.4, 0.0272783</pt>
   <pt>0.5, 0.0820396</pt>
```

```
      <pt>0.6, 0.201851</pt>
      <pt>0.7, 0.406524</pt>
      <pt>0.8, 0.670275</pt>
      <pt>0.9, 0.904792</pt>
      <pt>1.0, 0.999955</pt>
      <pt>1.1, 0.904792</pt>
      <pt>1.2, 0.670275</pt>
      <pt>1.3, 0.406524</pt>
      <pt>1.4, 0.201851</pt>
      <pt>1.5, 0.0820396</pt>
      <pt>1.6, 0.0272783</pt>
      <pt>1.7, 0.00740118</pt>
      <pt>1.8, 0.00161616</pt>
      <pt>1.9, 0.000258139</pt>
      <pt>2.0, 0</pt>
    </points>
    </Ftl>
  </discrete_material>
```

3.16.2 Discrete Section

After the discrete materials are defined, the materials are assigned to the discrete element sets that are defined in the *Mesh* section using the *discrete* element. This element requires two attributes:

dmat discrete material ID

discrete_set discrete element set defined in the *Mesh* section

3.16.3 Rigid Cable

A rigid cable can be used to apply a load to a series of rigid bodies that are connected at fixed points (fixed with respect to the rigid body). The cable runs through these points and the user can prescribe the force at the end of the cable. The rigid cable requires the following parameters.

force The magnitude of the force applied at the end of the cable.

force_direction The direction of the force (FEBio will normalize this vector if needed.)

relative If set to 1 the coordinates of the fixed points are relative to the rigid body frame of reference, otherwise they are in global coordinates.

point For each fixed point, enter the coordinates of the point. This tag requires the rb attribute to denote the rigid body it is attached to.

The following shows an example.

```
  <rigid_cable>
```

```
        <force lc="1">1000</force>
        <force_direction>0,0,-1</force_direction>
        <relative>1</relative>
        <point rb="1">0,0,0</point>
        <point rb="2">0,0,0</point>

    </rigid_cable>
```

This example defines a cable that runs through the centers of mass of two rigid bodies. The force is applied in the -z direction at the end of the cable (i.e. point 2).

3.17 Step Section

The analysis can be divided into separate steps, where in each step, different boundary and loading conditions can be applied. Each analysis step is defined via a *step* section. In each step, the following sections can be defined:

- Control
- Initial
- Boundary
- Loads
- Constraints
- Contact
- Rigid

When a boundary condition, (load, constraint, contact, etc.,) is applied to a step, then this boundary condition (load, constraint, contact, etc.) is only active during that step. Boundary conditions (loads, constraints, contacts, etc.) that are defined in the main body of the file will persist through all the step.

```
<!-- main body of file -->
 ...
<Step>
 <step id="1">
  <Control>
    ...
  </Control>
 </step>
 <step id="2">
   <Control>
    ...
   </Control>
 </step>
  ...
</Step>
```

3.18 LoadData Section

The *LoadData* section is used to define load controllers. A load controller allows users to manipulate the value of most model parameters as an explicit or implicit function of time. Use the *type* attribute to define a particular controller. Currently the following load controllers are available:

- **loadcurve**: the controller is a load curve, which is a tabulated list of time-value pairs. The user can set how the points are interpolated and how to extend the curve outside of its specified interval.
- **math**: the math controller allows the user to specify a mathematical expression that defines a function of time. Other model parameters can be used in the mathematical expression.
- **math-interval**: Similar to the math controller, but the math expression is defined only over a time interval. The behavior outside the interval can be specified separately.
- **PID**: a PID controller, which allows the value of a model parameter to be dependent on the values and their time history of other model parameters.

3.18.1 The loadcurve controller

A loadcurve controller is defined by the *loadcurve* type. Each loadcurve is defined by repeating the *point* element for all data points:

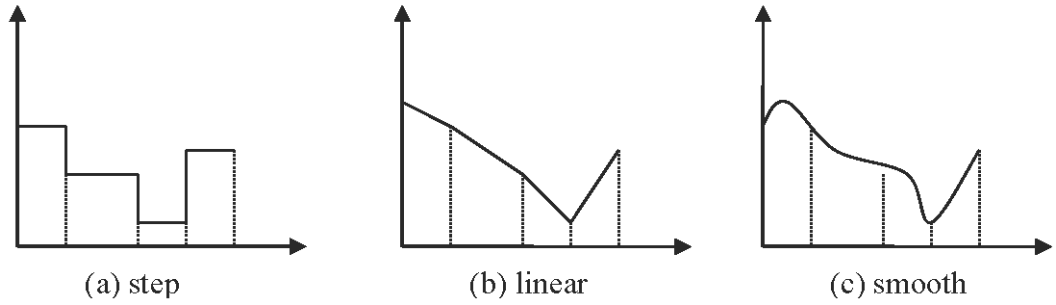
```
<load_controller id="1" type="loadcurve">
  <interpolate>LINEAR</interpolate>
  <extend>CONSTANT</extend>
  <points>
    <point> 0, 0 </point>
    ...
    <point> 1, 1 </point>
  </points>
</loadcurve>
```

For a loadcurve, the type is optional, since it is the default controller if the type attribute is omitted.

The *id* attribute is the loadcurve number and is used in other sections of the input file as a means to reference this curve.

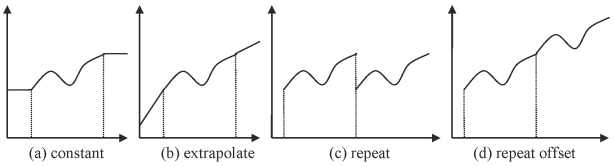
The optional parameters *interpolate* and *extend* define how the value of the loadcurve is interpolated from the data points. The interpolate defines the interpolation function and *extend* defines how the values of the loadcurve are determined outside of the interval defined by the first and last data point. The following tables list the possible values. The default values are marked with an asterisk (*).

interpolate	Description
STEP	Use a step interpolation function
LINEAR*	Use a linear interpolation function
SMOOTH	The values are interpolated using a cubic polynomial.



The different values for the *interpolate* parameter of load curves

Extend	Description
CONSTANT*	The value of the curve is the value of the closest endpoint
EXTRAPOLATE	The value is extrapolated linearly from the endpoints
REPEAT	The curve is repeated
REPEAT OFFSET	The curve is repeated but offset from the endpoints



The different values for the *extend* parameter of the load curve.

3.18.2 The math controller

Use the “math” value for the *type* attribute to define a math controller. This controller allows users to use a mathematical expression that defines a function of time. Use the “t” parameter to reference time.

```
<load_controller id="1" type="math">
    <math>2.0*sin(pi*t)</math>
</load_controller>
```

3.18.3 The PID controller

The PID controller allows users to create simple control systems where the value of one model parameter is used to control the output of another model parameter. The PID controller calculates the output value as a sum of three terms: a term proportional to the error (i.e. the difference between a user-defined target value and the measurement), a derivative term, and an integral term.

This controller requires the following parameters:

- **var**: specify the model parameter that is used as process variable, i.e. the measurement.
- **target**: the target value for the process variable.

- **Kp**: weight factor for the proportional term.
- **Kd**: weight factor for the derivative term
- **Ki**: weight factor for the integral term.

For example:

```
<load_controller id="1" type="PID">
    <var>fem.rigid_body[1].euler.z</var>
    <target>1.5708</target>
    <Kp>5</Kp>
    <Kd>1</Kd>
    <Ki>4</Ki>
</load_controller>
```

3.19 Output Section

FEBio usually splits the output in two files: the *logfile,* which contains the same information that was written to the screen during the analysis, and the *plotfile,* which contains the results. The contents of these output files can be customized in the *Output* section.

3.19.1 Logfile

The logfile records the same output that is printed to the screen. In addition, the user can request FEBio to output additional data to the logfile. This feature is called *data logging*. To use this feature, simply define the following element in the *Output* section of the input file:

```
<Output>
  <logfile [file="<log file>"]>
    <node_data [attributes]>item list</node_data>
    <element_data [attributes]>item list</element_data>
    <rigid_body_data [attributes]>item list</rigid_body_data>
  </logfile>
</Output>
```

The optional attribute *file* defines the name of the logfile. If omitted, a default name is used that is derived from the FEBio input file. See Section 2.7 for details on default naming conventions for output files.

Additional data is stored to the logfile by adding one or more of the following elements:

node_data request nodal data

face_data request surface data

element_data request element data

rigid_body_data request rigid body data

Each of these data classes takes the following attributes:

data an expression defining the data that is to be stored

name a descriptive name for the data (optional; default = data expression)

file the name of the output file where the data is stored. (optional; default = logfile)

delim the delimiter used to separate data in multi-column format (optional; default = space)

format an optional format string (optional; default = not used)

The *data* attribute is the most important one and is mandatory. It contains a list of variable names, separated by a semi-colon. The available variable names depend on the data class and are defined below. For example, the data expression:

```
data="x;y;z"
```

will store the variables x, y and z in separate columns. See below for more examples.

The optional *name* attribute is a descriptive name for the data. It is used in the logfile to refer to this data and can be used to quickly find the data record in the logfile. If omitted, the data expression is used as the name.

The *file* attribute defines the name of the output file where the data is to be stored. This attribute is optional and when not specified the data will be stored in the logfile.

The optional *delim* attribute defines the delimiter that is used in multi-column format. As described above, data can be stored in multiple columns and the delimiter is used to separate the columns. The default is a single space.

The optional *format* attribute defines a format string that will be used to format the output. If this attribute is present, the *delim* attribute will be ignored. The format string is composed of literal characters and special formatting characters. The special formatting characters are preceded by the percentage character (%). The following formatting characters are currently defined.

%i replace with the index of the corresponding item (i.e. node numbers for node data)

%g replace with a data value. Use a %g for each data item.

%t insert tab character in output.

%n insert newline character in output.

The following example,

```
<node_data data="x;y;z" format='<node id="%i">%g,%g,%g</node>'></node_data>
```

will print the following output (e.g. for node 1).

```
<node id="1">0.1,0.2,0.3</node>
```

Notice the use of the apostrophe (') in the format string. This is necessary in order to include the quotation marks as part of the format string. Also note that each data string will automatically be printed on a new line, so there is no need to end the format string with a newline character.

The value of the data elements is a list of items for which the data is to be stored. For example, for the *node_data* element the value is a list of nodes, for the *element_data* element it is a list of FE elements and for the *rigid_body* element it is a list of rigid bodies. The value may be omitted in which case the data for all items will be stored. For instance, omitting the value for the *node_data* element will store the data for all nodes.

As stated above, the data is either stored in the logfile or in a separate file. In any case, a record is made in the logfile. When storing the data in the logfile, the following entry will be found in the logfile at the end of each converged timestep for each data element:

```
Data Record #<n>
Step = <time step>
Time = <time value>
Data = <data name>
<actual data goes here>
```

The record number n corresponds to the nth data element in the input file. The *Step* value is the current time step. The *Time* value is the current solution time. The *Data* value is the name of the data element as provided by the *name* attribute (or the *data* attribute if *name* is omitted). The actual data immediately follows this record. If multiple column output is used, the columns are separated by the *delim* attribute of the data element.

When storing the data in a separate file, the format is slightly different:

```
Data Record #<n>
Step = <time step>
Time = <time value>
Data = <data name>
File = <file name>
```

The *File* value is the name of the physical file. Note that this is the name to which the time step number is appended. In addition, the physical file that stores the data contains the following header:

```
*Title = <problem title>
*Step = <time step>
*Time = <time value>
*Data = <data name>
<actual data goes here>
```

The problem title is as defined in the input file.

In either case, the actual data is a multi-column list, separated by the delimiter specified with the *delim* attribute (or a space when omitted). The first column always contains the item number. For example, the following data element:

```
<node_data
data="x;y;z" name="nodal coordinates" delim=",">1:4:1</node_data>
```

will result in the following record in the logfile:

```
Data Record #1
Step = 1
Time = 0.1
Data = "nodal coordinates"
1,0.000,0.000,0.000
2,1.000,0.000,0.000
3,1.000,1.000,0.000
4,0.000,1.000,0.000
```

This data record is repeated for each converged time step. The following sections define the data variables that are available for each of the data classes.

3.19.1.1 Node_Data Class

The *node_data* class defines a set of nodal variables. The data is stored for each node that is listed in the item list of the *node_data* element or for all nodes if no list is defined. The following nodal variables are defined.

Node variables	Description
x	x-coordinate of current nodal position
y	y-coordinate of current nodal position
z	z-coordinate of current nodal position
ux	x-coordinate of nodal displacement
uy	y-coordinate of nodal displacement
uz	z-coordinate of nodal displacement
vx	x-coordinate of nodal velocity
vy	y-coordinate of nodal velocity
vz	z-coordinate of nodal velocity
ax	x-coordinate of nodal acceleration
ay	y-coordinate of nodal acceleration
az	z-coordinate of nodal acceleration
Rx	x-coordinate of nodal reaction force
Ry	y-coordinate of nodal reaction force
Rz	z-coordinate of nodal reaction force

For analyses using biphasic, biphasic-solute, and triphasic materials, the following additional variables can be defined.

Node variables	Description
p	effective fluid pressure
vx	x-component of solid velocity
vy	y-component of solid velocity
vz	z-component of solid velocity
c[n]	effective concentration of solute n, with n from 1 to 8

For heat transfer problems the following nodal variables are available.

Node variables	Description
T	Nodal temperature

For fluid analyses, the following variables can be defined.

Node variables	Description
vx	x-component of fluid velocity
vy	y-component of fluid velocity
vz	z-component of fluid velocity

For example, to store the current nodal positions of all nodes, use the following *node_data* element:

```
<node_data data="x;y;z"></node_data>
```

You can store the total nodal displacement for nodes 1 through 100, and all even numbered nodes 200 through 400 as follows:

```
<node_data data="ux;uy;uz">1:100:1,200:400:2</node_data>
```

3.19.1.2 Face_Data Class

The *face_data* class defines a set of surface variables. This item requires the *surface* attribute, which defines the surface that will be used to extract the data from. The surface must be defined in the *Mesh* section. The data is stored for each surface facet that is listed in the item list, or for all facets of the surface if no list is defined.

The following surface variables are defined. Note that the actual value is the average over the facet's integration points values (if applicable).

Element variables	Description
contact gap	The gap (separation) of the contact interface this surface belongs to.
contact pressure	The normal contact pressure of the contact interface this surface belongs to.

Example:

This example stores the contact gap and contact pressure of all the facets of surface "FacetOnFacetSliding1_primary". The surface is defined in the Mesh section.

```
<logfile>
  <face_data data="contact gap;contact pressure" surface="FacetOnFacetSliding1_primary"/>
</logfile>
```

3.19.1.3 Element_Data Class

The *element_data* class defines a set of element variables. The data is stored for each element that is listed in the item list of the *element_data* element or for all nodes if no list is defined. The following element variables are defined. Note that the actual value is the average over the element's integration points values (if applicable).

Element variables	Description
x	x-coordinate of current element centroid position
y	y-coordinate of current element centroid position
z	z-coordinate of current element centroid position
sx	xx-component of the Cauchy stress
sy	yy-component of the Cauchy stress
sz	zz-component of the Cauchy stress
sxy	xy-component of the Cauchy stress
syz	yz-component of the Cauchy stress
sxz	xz-component of the Cauchy stress
s1	first eigenvalue of Cauchy stress tensor
s2	second eigenvalue of Cauchy stress tensor
s3	third eigenvalue of Cauchy stress tensor
Ex	xx-component of the Green-Lagrange strain
Ey	yy-component of the Green-Lagrange strain

Ez	zz-component of the Green-Lagrange strain
Exy	xy-component of the Green-Lagrange strain
Eyz	yz-component of the Green-Lagrange strain
Exz	xz-component of the Green-Lagrange strain
E1	first eigenvalue of Green-Lagrange strain tensor
E2	second eigenvalue of Green-Lagrange strain tensor
E3	third eigenvalue of Green-Lagrange strain tensor
ex	xx-component of the infinitesimal strain
ey	yy-component of the infinitesimal strain
ez	zz-component of the infinitesimal strain
exy	xy-component of the infinitesimal strain
eyz	yz-component of the infinitesimal strain
exz	xz-component of the infinitesimal strain
Ux	xx-component of the right stretch tensor
Uy	yy-component of the right stretch tensor
Uz	zz-component of the right stretch tensor
Uxy	xy-component of the right stretch tensor
Uyz	yz-component of the right stretch tensor
Uxz	xz-component of the right stretch tensor
U1	first eigenvalue of right stretch tensor
U2	second eigenvalue of right stretch tensor
U3	third eigenvalue of right stretch tensor
Vx	xx-component of the left stretch tensor
Vy	yy-component of the left stretch tensor
Vz	zz-component of the left stretch tensor
Vxy	xy-component of the left stretch tensor
Vyz	yz-component of the left stretch tensor
Vxz	xz-component of the left stretch tensor
V1	first eigenvalue of left stretch tensor
V2	second eigenvalue of left stretch tensor
V3	third eigenvalue of left stretch tensor
Hx	xx-component of the right Hencky strain
Hy	yy-component of the right Hencky strain
Hz	zz-component of the right Hencky strain
Hxy	xy-component of the right Hencky strain
Hyz	yz-component of the right Hencky strain
Hxz	xz-component of the right Hencky strain
H1	first eigenvalue of right Hencky strain
H2	second eigenvalue of right Hencky strain
H3	third eigenvalue of right Hencky strain
hx	xx-component of the left Hencky strain
hy	yy-component of the left Hencky strain
hz	zz-component of the left Hencky strain
hxy	xy-component of the left Hencky strain
hyz	yz-component of the rileft Hencky strain

hxz	xz-component of the left Hencky strain
h1	first eigenvalue of left Hencky strain
h2	second eigenvalue of left Hencky strain
h3	third eigenvalue of left Hencky strain
Fxx	xx-component of the deformation gradient
Fyy	yy-component of the deformation gradient
Fzz	zz-component of the deformation gradient
Fxy	xy-component of the deformation gradient
Fyz	yz-component of the deformation gradient
Fxz	xz-component of the deformation gradient
Fyx	yx-component of the deformation gradient
Fzy	zy-component of the deformation gradient
Fzx	xz-component of the deformation gradient
J	relative volume (determinant of deformation gradient)
cxxxx	xxxx component of spatial elasticity tensor (a.k.a. c11)
cxxyy	xxyy component of spatial elasticity tensor (a.k.a. c12)
cyyyy	yyyy component of spatial elasticity tensor (a.k.a. c22)
cxxzz	xxzz component of spatial elasticity tensor (a.k.a. c13)
cyyzz	yyzz component of spatial elasticity tensor (a.k.a. c23)
czzzz	zzzz component of spatial elasticity tensor (a.k.a. c33)
cxxxy	xxxy component of spatial elasticity tensor (a.k.a. c14)
cyyxy	yyxy component of spatial elasticity tensor (a.k.a. c24)
czzxy	zzxy component of spatial elasticity tensor (a.k.a. c34)
cxyxy	xyxy component of spatial elasticity tensor (a.k.a. c44)
cxxyz	xxyz component of spatial elasticity tensor (a.k.a. c15)
cyyyz	yyyz component of spatial elasticity tensor (a.k.a. c25)
czzyz	zzyz component of spatial elasticity tensor (a.k.a. c35)
cxyyz	xyyz component of spatial elasticity tensor (a.k.a. c45)
cyzyz	yzyz component of spatial elasticity tensor (a.k.a. c55)
cxxxz	xxxz component of spatial elasticity tensor (a.k.a. c16)
cyyxz	yyxz component of spatial elasticity tensor (a.k.a. c26)
czzxz	zzxz component of spatial elasticity tensor (a.k.a. c36)
cxyxz	xyxz component of spatial elasticity tensor (a.k.a. c46)
cyzyz	yzyz component of spatial elasticity tensor (a.k.a. c56)
cxzxz	xzxz component of spatial elasticity tensor (a.k.a. c66)
sed	strain energy density
devsed	deviatoric strain energy density
D	damage variable (or sum of damage variables in solid mixtures)

For analyses using biphasic, biphasic-solute, and triphasic materials, the following additional variables can be defined:

Element variables	**Description**

p	actual fluid pressure
cn	actual concentration of solute n
wx	x-component of fluid flux
wy	y-component of fluid flux
wz	z-component of fluid flux
jnx	x-component of flux of solute n
jny	y-component of flux of solute n
jnz	z-component of flux of solute n

For fluid analyses, the following additional variables can be defined:

Element variables	**Description**
fx	x-coordinate of element centroid position
fy	y-coordinate of element centroid position
fz	z--coordinate of element centroid position
fp	fluid pressure
fvx	x-component of fluid velocity
fvy	y-component of fluid velocity
fvz	z-component of fluid velocity
fJ	fluid volume ratio
fd	fluid density
fsp	fluid stress power
fax	x-component of fluid acceleration
fay	y-component of fluid acceleration
faz	z-component of fluid acceleration
fwx	x-component of fluid vorticity
fwy	y-component of fluid vorticity
fwz	z-component of fluid vorticity
fsxx	xx-component of fluid Cauchy stress
fsyy	yy-component of fluid Cauchy stress
fszz	zz-component of fluid Cauchy stress
fsxy	xy-component of fluid Cauchy stress
fsyz	yz-component of fluid Cauchy stress
fsxz	xz-component of fluid Cauchy stress
fdxx	xx-component of fluid rate of deformation
fdyy	yy-component of fluid rate of deformation
fdzz	zz-component of fluid rate of deformation
fdxy	xy-component of fluid rate of deformation
fdyz	yz-component of fluid rate of deformation
fdxz	xz-component of fluid rate of deformation

For example, to store the (average) Cauchy stress for all elements, define the following data element:

```
<element_data data="sx;sy;sz;sxy;syz;sxz" name="element stresses"> </element_data>
```

3.19.1.4 Rigid_Body_Data Class

The *rigid_body_data* class defines a set of variables for each rigid body. The data is stored for each rigid body that is listed in the item list of the *rigid_body_data* element or for all rigid bodies if no list is defined. The following variables are defined. Note that the item referenced in the item list is the material number of the rigid body.

Rigid body variables	Description
x	x-coordinate of center of mass position
y	y-coordinate of center of mass position
z	z-coordinate of center of mass position
vx	x-component of center of mass velocity
vy	y-component of center of mass velocity
vz	z-component of center of mass velocity
ax	x-component of center of mass acceleration
ay	y-component of center of mass acceleration
az	z-component of center of mass acceleration
thx	x-component of rotation pseudo-vector
thy	y-component of rotation pseudo-vector
thz	z-component of rotation pseudo-vector
omx	x-component of angular velocity
omy	y-component of angular velocity
omz	z-component of angular velocity
alx	x-component of angular acceleration
aly	y-component of angular acceleration
alz	z-component of angular acceleration
qx	x-component of rotation quaternion
qy	y-component of rotation quaternion
qz	z-component of rotation quaternion
qw	w-component of rotation quaternion
Fx	x-component of the rigid body reaction force
Fy	y-component of the rigid body reaction force
Fz	z-component of the rigid body reaction force
Mx	x-component of the rigid body reaction torque
My	y-component of the rigid body reaction torque
Mz	z-component of the rigid body reaction torque
KE	kinetic energy

For example, to store the rigid body reaction force of rigid body 2 and 4 add the following data element. Note that the 2 and 4 refer to the rigid body material number as defined in the *Material* section of the input file:

```
<rigid_body_data data="Fx;Fy;Fz">2,4</rigid_body_data>
```

3.19.1.5 Rigid_Connector_Data Class

The *rigid_connector_data* class defines a set of variables for each rigid joint or rigid connector. The data is stored for each rigid joint or rigid connector that is listed in the item list of the *rigid_connector_data* element or for all rigid connectors if no list is defined. The following variables are defined. Note that the item referenced in the item list is the rigid connector number in the order in which rigid connectors appear in the input file.

Rigid body variables	Description
RCFx	x-component of rigid connector force
RCFy	y-component of rigid connector force
RCFz	z-component of rigid connector force
RCMx	x-component of rigid connector moment
RCMy	y-component of rigid connector moment
RCMz	z-component of rigid connector moment
RCx	x-component of rigid connector translation
RCy	y-component of rigid connector translation
RCz	z-component of rigid connector translation
RCthx	x-component of rigid connector rotation
RCthy	y-component of rigid connector rotation
RCthz	z-component of rigid connector rotation

For example, to store the reaction forces and moments at rigid joints 2 and 4 add the following data element:

```
<rigid_connector_data data="RCFx;RCFy;RCFz;RCMx;RCMy;RCMz">2,4</rigid_connector_data>
```

The rigid connector translation and rotation variables return relative motions of the rigid bodies connected by rigid joints and other rigid connectors (Section 3.12.4). The rotation components represent the components of a vector whose direction is along the axis of rotation, and whose magnitude is the (counter-clockwise) angle of rotation. These relative motions are calculated as the motion of *body_b* relative to *body_a*, expressed in the local Cartesian basis of *body_a*, whose initial origin is at *joint_origin*. This Cartesian basis (which generally moves with *body_a*) has its first axis along *rotation_axis* (for revolute and planar joints) or *translation_axis* (for prismatic joints) or *joint_axis* (for cylindrical joints); the second axis is along the *transverse_axis* (for revolute, prismatic and cylindrical joints) or *translation_axis_1* (for planar joints). For spherical joints, springs, dampers, angular dampers and contractile forces, the axes are always aligned with the global Cartesian basis.

For springs, dampers and contractile forces, the returned translation components represent the relative position vector between insertion points on *body_b* and *body_a*. The magnitude of this vector represents the distance between those points.

3.19.2 Plotfile

By default, all the results are stored in a binary database, referred to as the *plotfile*. The preferred storage format is the FEBio binary database format (referred to as the *xplt* format)[4]. This section describes how to customize the data that is stored in the *xplt* format.

To define the contents of the plotfile the *plotfile* element needs to be added in the *Output* section of the FEBio input file.

```
<plotfile [type="febio"] [file="name.xplt"]/>
```

The *type* attribute defines the output format. Currently, only "febio" is supported, and thus defining the type attribute is optional.

The *file* attribute is also optional and allows the user to define the file name of the plotfile. If this attribute is omitted, FEBio will use a default file name for the plotfile.

By default, FEBio will store the most common data variables to the plot file. However, it is advised to always specify the specific contents of the plotfile. This can be done by adding *var* elements in the *plotfile* section as described in section 3.19.2.1.

3.19.2.1 Plotfile Variables

Plotfile variables are defined using the *var* keyword. This tag takes one attribute, namely the *type* of the variable. The *type* defines the specific output variable that will be stored to the plot file.

```
<var type="name"/>
```

where *name* is the name of the output variable. (For a complete list of supported output variables, please see section 3.19.2.2)

As of FEBio 2.4, additional information can be added in the type attribute of the variable definition. The general format is as follows.

```
<var type="name[filter]=alias"/>
```

The filter is defined by appending the name of the plot variable with the filter inside square brackets. The filter can be either an integer, or a string. If the filter is a string, it must be surrounded by single quotes.

```
<var type="name['filter string']=alias"/>
```

The optional alias can be used to rename the variable. The alias will be used as the name of the plot variable in the plot file, and this is the name that will show up in post-processing software such as FEBio Studio.

Some plot variables use filters to resolve possible ambiguities. For example,

```
<var type="solute concentration['solute1']"/>
```

This example will store the solute concentration of a solute named 'solute1' to the plot file.

In addition, an alias can be used to define a more descriptive name of the plot variable.

[4] As of FEBio version 2.0, the LSDYNA database is no longer supported. The FEBio database format is the only format that will be supported from now on.

```
<var type="solute concentration['Na']=Na concentration"/>
```

This variable will store the solute concentration of a solute named 'Na' to the plot file using the name 'Na concentration'.

The following example shows how to export a particular fiber vector from a solid mixture. Since solid mixtures can have several fiber distributions associated, it is necessary to specify which component of the mixture the plot variable refers to.

```
<var type="fiber vector['solid[0]']=v1"/>
<var type="fiber vector['solid[1]']=v2"/>
```

Some plot variables require the specification of a named surface (* in table below). For example,

```
<var type="fluid surface force" surface="airfoil"/>
```

The named surface should be described in a *Surface* element (Section 3.6.5) within *Mesh* (Section 3.6).

The following example shows a complete definition of the plotfile section and stores nodal displacements and element stresses.

Example:

```
<plotfile type="febio">
  <var type="displacement"/>
  <var type="stress"/>
</plotfile>
```

3.19.2.2 List of Plotfile Variables

There are four categories of plot file variables:

- **Nodal:** These typically correspond to data that is directly available at nodes. For instance, the primary solution variables are usually stored as nodal variables. (E.g. "displacement")
- **Element:** These quantities correspond to data that is typically stored on the elements (e.g. stress). In FEBio, element data is usually stored at the integration points during the solution phase. Since this is often not a convenient format for post-processing, element data is processed before export and will be written to the plot file as either a constant, element-averaged value, or projected to the nodes of the element.
- **Surface:** These quantities correspond to data that is only available at the surface of the mesh (e.g. contact traction). If the surface data is defined at the nodes of the surface, it will be written directly to the plot file. However, if the data is associated with the surface facets - or more accurately, with the integration points of the surface facet - then it will be processed before export, and a constant, facet-averaged value, will be written to the plot file.
- **Rigid body:** These are quantities that correspond to data stored on rigid bodies.

The following table lists the possible values for the *type* attribute of the plot file variable tag.

An asterisk (*) in the Category column denotes that the attribute is required and must be included in the definition of the plot variable.

Type	Category	Description
Acceleration	Element	Average element accelerations
contact area	Surface	Net contact area across contact interface. Evaluated by integrating the surface area at integration points where the contact pressure is not zero.
contact force	Surface	Net contact force across contact interface. Evaluated by integrating the contact traction over the contact surface.
contact gap	Surface	Contact gap distance, evaluated at contact surface faces
contact penalty	Surface	Contact penalty value
contact pressure	Surface	Contact pressure, evaluated at contact surface faces
contact stick	Surface	Stick status in stick-slip frictional contact (1=stick, 0=slip). Values between 0 and 1 imply that a fraction of that face is in stick mode. Currently works only with sliding-tension-compression.
contact traction	Surface	Contact traction vectors, evaluated at contact surface faces
current density	Element	Electric current density, Eq.(4.12.11)
current element angular momentum	Element	The element's total angular momentum at the current configuration.
current element center of mass	Element	The element's center of mass at the current configuration.
current element kinetic energy	Element	The element's total kinetic energy at the current configuration.
current element linear momentum	Element	The element's total linear momentum at the current configuration
current element strain energy		The element's total strain energy at the current configuration.
damage	Element	Damage variable D (Section 4.6)
density	Element	The current density
deviatoric strain energy density	Element	Deviatoric strain energy density $\tilde{\Psi}\left(\tilde{\mathbf{C}}\right)$, Eq.(4.1.1)
displacement	Node	Nodal displacements
effective elasticity	Element	The elasticity of the elastic material in a biphasic/triphasic/multiphasic material.
effective fluid pressure	Node	fluid pressure for biphasic, biphasic-solute, triphasic, and multiphasic materials.
effective shell fluid pressure	Node	fluid pressure for biphasic, biphasic-solute, triphasic, and multiphasic materials of shell domains.

effective shell solute concentration	Node	Effective solute concentration for biphasic-solute, triphasic, and multiphasic materials of shell domains.
effective solute concentration	Node	Effective solute concentration for biphasic-solute, triphasic, and multiphasic materials.
effective fluid pressure	Node	Fluid pressure p for biphasic materials, Eq.(4.10.1), or $\tilde{p}$ for biphasic-solute, Eq.(4.11.3) and triphasic/multiphasic materials, Eq.(4.12.7)
effective friction coefficient	Surface	Effective friction coefficient at a biphasic contact interface
effective solute concentration	Node	Effective solute concentration $\tilde{c}$ for biphasic-solute materials, Eq.(4.11.3), and triphasic/multiphasic materials, Eq.(4.12.7)
elasticity	Element	Spatial elasticity tensor components
electric potential	Element	Electric potential ψ in triphasic/multiphasic materials, Section 4.12
element angular momentum	Element	Element total angular momentum
element center of mass	Element	Element center of mass
element kinetic energy	Element	Element total kinetic energy
element linear momentum	Element	Element total linear momentum
element strain energy	Element	Element total strain energy
element stress power	Element	Element total stress power
enclosed volume	Surface*	Volume enclosed by named surface
Euler angle	Rigid body	Rigid body Euler angles
fiber stretch	Element	Element's average fiber stretch
fiber vector	Element	Material fiber vector
fixed charge density	Element	Fixed charge density c^F in current configuration, Eq.(4.12.2)
fluid acceleration	Element	Fluid acceleration (material time derivative of fluid velocity $\mathbf{v}^f$, Section (4.16)
fluid density	Element	Fluid density ρ^f in fluid and fluid-FSI materials, Eq.(4.16.4)
fluid dilatation	Node	Fluid dilatation e, Section (4.16))
fluid element angular momentum	Element	Total angular momentum of fluid element
fluid element center of mass	Element	Center of mass of fluid element
fluid element kinetic energy	Element	Total kinetic energy of fluid element
fluid element linear momentum	Element	Total linear momentum of fluid element
fluid element strain energy	Element	Total strain energy of fluid element
fluid energy density	Element	Average energy density of fluid element
fluid flow rate	Surface*	Volumetric fluid flow rate $Q = \int_A \mathbf{w} \cdot \mathbf{n}\, dA$ across a named surface (biphasic and multiphasic analyses)

fluid flux	Element	Fluid flux $\mathbf{w}$ in biphasic, Eq.(4.10.2), biphasic-solute, Eq.(4.11.4), triphasic/multiphasic, Eq.(4.12.9), fluid-FSI and biphasic-FSI materials.
fluid force	Surface	Net fluid force across biphasic (sliding-biphasic), biphasic-solute (sliding-biphasic-solute) and multiphasic (sliding-multiphasic) contact interface. Evaluated by integrating the fluid pressure p over the contact surface.
fluid force2	Surface	Alternative calculation of the net fluid force
fluid heat supply density	Element	Average heat supply density of a fluid element
fluid kinetic energy density	Element	Average kinetic energy densify of a fluid element
fluid load support	Surface	The total fluid load support across a named surface
fluid mass flow rate	Surface*	Mass flow rate across a named surface, $\dot{m} = \int_A \rho \mathbf{v} \cdot \mathbf{n}\, dA$, Section (4.16)
fluid pressure	Element	Fluid pressure p in biphasic, Eq.(4.10.1), biphasic-solute, Eq.(4.11.3), triphasic/multiphasic materials, Eq.(4.12.7), and fluid materials, Eq.(**??**)
fluid rate of deformation	Element	Fluid rate of deformation tensor $\mathbf{D}$, Section (4.16)
fluid shear viscosity	Element	Average shear viscosity of a fluid element
fluid strain energy density	Element	Average shear energy density of fluid element
fluid stress	Element	Fluid stress $\boldsymbol{\sigma}$, Eq.(4.16.1)
fluid stress power density	Element	Fluid stress power $\boldsymbol{\sigma} : \mathbf{D}$, Section (4.16)
fluid surface energy flux	Surface	The total energy flux across a named surface for a fluid analysis
fluid surface force	Surface*	Net force exerted by a fluid on a named surface, $\mathbf{f} = -\int_A \mathbf{t}\, dA$, Section (4.16)
fluid surface pressure	Surface*	Average fluid pressure on each face of a surface, evaluated from the nodal values of the fluid dilatation on that face
fluid surface traction power	Surface	Net traction power across a named surface of a fluid analysis
fluid velocity	Element	Element fluid velocity $\mathbf{v}^f$ in fluid and fluid-FSI materials, Section (4.16)
fluid volume ratio	Element	Fluid volume ratio J, Section (4.16)
fluid vorticity	Element	Fluid vorticity, Section (4.16)
heat flux	Element	The average heat flux inside an element of a heat-transfer analysis

kinetic energy density	Element	The average kinetic energy density in an element
Lagrange strain	Element	The average Lagrange strain in an element
left Hencky	Element	Left Hencky strain $\mathbf{h} = \frac{1}{2}\ln\mathbf{b}$, where $\mathbf{b} = \mathbf{F}\cdot\mathbf{F}^T$
left stretch	Element	Left stretch tensor $\mathbf{V} = \mathbf{b}^{1/2}$
local fluid load support	Surface	Local fluid load support on biphasic contact surfaces
nested damage	Element	The average damage in an elastic mixture
nodal acceleration	Node	The acceleration at the nodes
nodal contact gap	Surface	Contact gap distance, evaluated at contact surface nodes
nodal contact pressure	Surface	Contact pressure, evaluated at contact surface nodes
nodal contact traction	Surface	Contact traction vectors, evaluated at contact surface nodes
nodal fluid flux	Node	Fluid flux in biphasic, biphasic-solute, triphasic, multiphasic, fluid-FSI and biphasic-FSI materials
nodal fluid velocity	Node	Fluid velocity $\mathbf{v}^f$ in fluid and fluid-FSI materials
nodal stress	Element	The Cauchy stress, projected to the nodes.
nodal surface traction	Surface	Contact nodal tractions
nodal vector gap	Surface	Contact gap vector at surface nodes
nodal velocity	Node	The nodal velocities
osmolarity	Element	Element's average osmolarity in biphasic/triphasic/multiphasic analysis
parameter	Element	Stores heterogeneous material parameter data
pressure gap	Surface	Pressure gap between opposing surfaces in a biphasic contact analysis
reaction forces	Node	The nodal reaction forces.
receptor-ligand concentration	Element	Receptor-ligand concentration in a biphasic-solute analysis.
referential fixed charge density	Element	Referential fixed charge density c_r^F, Eq.(4.12.2), which may evolve with chemical reactions, Eq.(4.13.10)
referential solid volume fraction	Element	Referential solid volume fraction φ_r^s, which may evolve with chemical reactions, Eq.(4.13.9)
relative fluid velocity	Element	Average element fluid velocity relative to mesh $\mathbf{w}$ in fluid and fluid-FSI materials
relative volume	Element	Relative volume $J = \det\mathbf{F}$
right Hencky	Element	Right Hencky strain $\mathbf{H} = \frac{1}{2}\ln\mathbf{C}$
right stretch	Element	Right stretch tensor $\mathbf{U} = \mathbf{C}^{1/2}$
rigid acceleration	Rigid body	Rigid body center of mass acceleration

rigid angular acceleration	Rigid body	Rigid body angular acceleration
rigid angular momentum	Rigid body	Rigid body angular momentum
rigid angular position	Rigid body	Rigid body rotation pseudo-vector
rigid angular velocity	Rigid body	Rigid body angular velocity
rigid force	Rigid body	Total force applied to the rigid body
rigid kinetic energy	Rigid body	Rigid body kinetic energy
rigid linear momentum	Rigid body	Rigid body linear momentum
rigid position	Rigid body	Rigid body center of mass position
rigid rotation vector	Rigid body	Rigid body rotation pseudo-vector
rigid torque	Rigid body	Rigid body moment
rigid velocity	Rigid body	Rigid body center of mass velocity
RVE generations	Element	Number of generations in reactive viscoelastic material
RVE reforming bonds	Element	Fraction of weak bonds available to break and reform in reactive viscoelastic material
RVE strain	Element	Strain measure used in strain-dependent reactive viscoelastic relaxation and weak bond recruitment
sbm concentration	Element	Average element solid-bound molecule concentration
sbm referential apparent density	Element	Average element solid-bound molecule referential apparent density
shell director	Element	The shell director
shell relative volume	Element	Relative volume of shell element
shell strain	Element	Average strain in shell element
shell thickness	Node	Shell thickness
solute concentration	Element	Solute concentration c in biphasic-solute materials, or c^α in triphasic/multiphasic materials
solute flux	Element	Solute flux $\mathbf{j}$ in biphasic-solute materials, Eq.(4.11.4), or $\mathbf{j}^\alpha$ in triphasic/multiphasic materials, Eq.(4.12.9).
specific strain energy	Element	Average element specific strain energy
SPR principal stress	Element	Principal stress values obtained via SPR projection
SPR stress	Element	Cauchy stress obtained via SPR projection
SPR-P1 stress	Element	Cauchy stress obtained via SPR projection (first-order projection)
strain energy density	Element	Strain energy density $\Psi(\mathbf{C})$
stress	Element	Cauchy stress
surface traction	Surface	Nodal surface tractions of nodes on a contact surface
uncoupled pressure	Element	The pressure of an uncoupled material
ut4 nodal stress	Element	Stresses at the nodes of the UT4 element
vector gap	Element	The vector gap at nodes of a contact surface

velocity	Node	Nodal velocities (fluid nodal velocities in fluid analyses)
volume fraction	Element	Average element volume fraction of a solid mixture
in-situ target stretch	Element	the fiber stretch induced by the initial prestrain gradient
prestrain stretch	Element	the stretch induced by the effective prestrain gradient
prestrain correction	Element	the 3x3 prestrain correction tensor
SPR prestrain correction	Element	The 3x3 prestrain correction tensor (recovered with SPR)

Chapter 4

Materials

The following sections describe the material parameters for each of the available constitutive models, along with a short description of each material. A more detailed theoretical description of the constitutive models can be found in the *FEBio Theory Manual*.

4.1 Elastic Solids

This section describes the elastic materials, which are materials defined by a hyperelastic strain-energy function. A distinction will be made between so-called *unconstrained* and *uncoupled* materials. The former describe materials that can undergo volumetric compression. The latter are used for modeling (nearly-) incompressible materials, thus their constitutive models are constrained to produce an isochoric deformation, as long as the user has selected a sufficiently large bulk modulus to enforce that constraint.

4.1.1 Specifying Fiber Orientation or Material Axes

Some of the materials are transversely isotropic, requiring the specification of an initial material direction, which is called a *fiber* direction in FEBio. Other materials are orthotropic, requiring the specification of initial material axes that define the three planes of symmetry for those materials. Only one of these specifications should be provided. By default, material axes are aligned with the global Cartesian basis $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ and the fiber direction is along $\mathbf{e}_1$. Local fiber or material axes orientation may be specified in several ways. FEBio gives the option to automatically generate the orientation, based on some user-specified parameters. However, the user can override this feature and specify the fiber or axes directions for each element manually in the *ElementData* section. See Section 3.8 for more details on how to do this.

4.1.1.1 Transversely Isotropic Materials

For transversely isotropic materials fiber orientation is specified with the *fiber* element. This element takes one attribute, namely *type*, which specifies the type of the fiber generator. The possible values are specified in the following table.

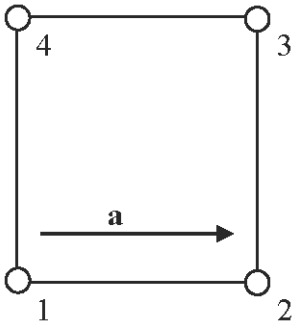
***type* Value**	**Description**
local	Use local element numbering (1)
vector	Define the fiber orientation as a constant vector (2)
spherical	specifies a spherical fiber distribution (3)
cylindrical	specifies a cylindrical fiber distribution (4)
angles	Specifies the fiber direction using spherical angles (5)

If the *type* attribute is omitted, the fiber distribution will follow the local element nodes 1 and 2. This would be the same as setting the fiber attribute to *local* and setting the value to "1,2".

Comments:

1. In this case, the fiber direction is determined by local element node numbering. The value is interpreted as the local node numbers of the nodes that define the direction of the fiber. The following example defines a local fiber axis by local element nodes 1 and 2. This option is very useful when the local fiber direction can be interpreted as following one of the mesh edges.

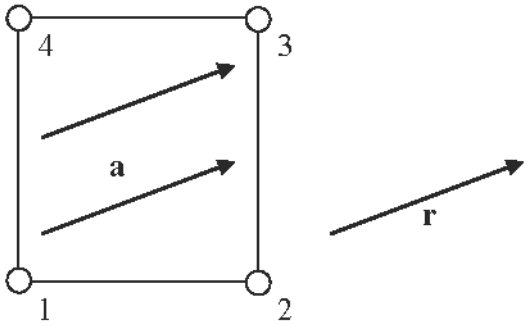
```
<fiber type="local">1,2</fiber>
```



***local* fiber direction option**

2. The fiber orientation is specified by a vector. The value is the direction of the fiber. The following defines all element fiber directions in the direction of the vector [1,0,0]:

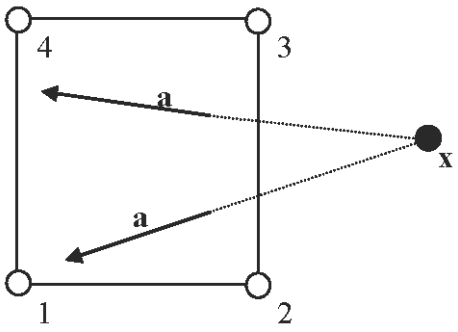
```
<fiber type="vector">1,0,0</fiber>
```



***vector* fiber direction option**

3. The fiber orientation is determined by a point in space and the global location of each element integration point. The value is the location of the point. The following example defines a spherical fiber distribution centered at [0,0,1]:

```
<fiber type="spherical">0,0,1</fiber>
```



***spherical* fiber direction option**

4. *cylindrical*: This type generates a fiber distribution that is cylindrical. The following subparameters must be defined.

   center defines the center of the cylinder

   axis defines the axis of the cylinder

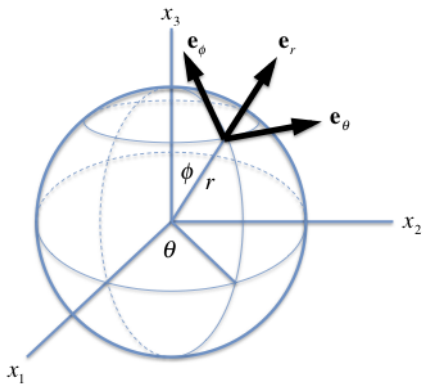
   vector defines a vector that will be transported around the cylinder

```
<fiber type="cylindrical">
  <center>0,0,0</center>
  <axis>0,0,1</axis>
  <vector>0,1,0</vector>
</fiber>
```

5. *angles*: This type generates a fiber orientation via the specification of spherical angles (azimuth and declination) relative to the local material axes (or global coordinate system, if no local material axes are defined). The following subparameters must be defined.

   theta azimuth angle (in degrees)

   phi declination angle (in degrees)



Spherical angles

The fiber is oriented along

$$\mathbf{e}_r = \cos\theta\sin\phi\,\mathbf{e}_1 + \sin\theta\sin\phi\,\mathbf{e}_2 + \cos\phi\,\mathbf{e}_3, 0 \leqslant \theta < 2\pi,\ 0 \leqslant \phi \leqslant \pi,$$

where $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ are orthonormal vectors representing the local element coordinate system (when specified, Section 4.1.1.2), or global coordinate system.

```
<fiber type="angles">
  <theta>20</center>
  <phi>90</phi>
</fiber>
```

When specifying a fiber direction $\mathbf{a}$, FEBio generates a set of orthogonal material axes as described in Section 4.1.1.2. generated with $\mathbf{d} = \boldsymbol{j}$, or else $\mathbf{d} = \boldsymbol{k}$ if $\mathbf{a}$ is collinear with $\boldsymbol{j}$ (where the triple $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ refers to unit vectors defining the global coordinate system, i.e. $\boldsymbol{i} = (1, 0, 0)$, etc.). Because of the non-uniqueness of these material axes (only $\mathbf{e}_1$ is along a uniquely defined direction in a *fiber* element), caution should be used when material axes are compounded, as may occur in nested materials such as solid mixtures described in Sections 4.1.2.16 & 4.1.4.26. To enforce uniqueness, use the *mat_axis* element instead of the *fiber* element.

4.1.1.2 Orthotropic Materials

For orthotropic materials, the user needs to specify two fiber directions $\mathbf{a}$ and $\mathbf{d}$. From these FEBio will generate an orthonormal set of material axes vectors as follows:

$$\mathbf{e}_1 = \frac{\mathbf{a}}{\|\mathbf{a}\|}, \quad \mathbf{e}_2 = \mathbf{e}_3 \times \mathbf{e}_1, \quad \mathbf{e}_3 = \frac{\mathbf{a} \times \mathbf{d}}{\|\mathbf{a} \times \mathbf{d}\|}.$$

The vectors $\mathbf{a}$ and $\mathbf{d}$ are defined using the *mat_axis* element. This element takes a *type* attribute, which can take on the following values:

Value	Description
local	Use local element numbering (1)
vector	Specify the vectors $\mathbf{a}$ and $\mathbf{d}$ directly. (2)
angles	Specify the angles θ and φ [deg]. (3)

Comments:

1. When specifying *local* as the material axis type, the value is interpreted as a list of three local element node numbers. When specifying zero for all three, the default (1,2,4) is used.

```
<mat_axis type="local">0,0,0</mat_axis>
```

2. When using the *vector* type, you need to define the two generator vectors a and d. These are specified as child elements of the *mat_axis* element:

```
<mat_axis type="vector">
  <a>1,0,0</a>
  <d>0,1,0</d>
</mat_axis>
```

3. When using the *angle* type, you need to define the two angles θ and φ in degrees. These are specified as child elements of the *mat_axis* element:

```
<mat_axis type="angles">
  <theta>0</theta>
  <phi>90</phi>
</mat_axis>
```

The material axes $\{\mathbf{e}_r, \mathbf{e}_\theta, \mathbf{e}_\varphi\}$ are related to the global Cartesian basis (or local element axes) $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ via

$$\begin{aligned} \mathbf{e}_r &= \cos\theta \sin\phi\, \mathbf{e}_1 + \sin\theta \sin\phi\, \mathbf{e}_2 + \cos\phi\, \mathbf{e}_3 \\ \mathbf{e}_\theta &= -\sin\theta\, \mathbf{e}_1 + \cos\theta\, \mathbf{e}_2 \\ \mathbf{e}_\varphi &= -\cos\theta \cos\phi\, \mathbf{e}_1 - \sin\theta \cos\phi\, \mathbf{e}_2 + \sin\phi\, \mathbf{e}_3 \end{aligned} \quad .$$

4.1.2 Uncoupled Materials

Uncoupled, nearly-incompressible hyperelastic materials are described by a strain energy function that features an additive decomposition of the hyperelastic strain energy into deviatoric and dilational parts [73]:

$$\Psi\left(\mathbf{C}\right) = \tilde{\Psi}\left(\tilde{\mathbf{C}}\right) + U\left(J\right), \tag{4.1.1}$$

where $\tilde{\mathbf{C}} = \tilde{\mathbf{F}} \cdot \tilde{\mathbf{F}}^T$ and $\tilde{\mathbf{F}} = J^{-1/3}\mathbf{F}$ is the deviatoric part of the deformation gradient. The resulting 2^{nd} Piola-Kirchhoff stress is given by

$$\mathbf{S} = J^{-2/3}\,\mathrm{Dev}\left[\tilde{\mathbf{S}}\right] + pJ\mathbf{C}^{-1}, \tag{4.1.2}$$

where

$$\tilde{\mathbf{S}} = 2\frac{\partial\tilde{\Psi}}{\partial\tilde{\mathbf{C}}} \quad , \tag{4.1.3}$$

and

$$p := \frac{dU}{dJ}, \tag{4.1.4}$$

and $\mathrm{Dev}\left[\cdot\right]$ is the deviatoric operator in the material frame.

The corresponding Cauchy stress is given by

$$\boldsymbol{\sigma} = \mathrm{dev}\left[\tilde{\boldsymbol{\sigma}}\right] + p\mathbf{I}, \tag{4.1.5}$$

where $\tilde{\boldsymbol{\sigma}} = J^{-1}\tilde{\mathbf{F}} \cdot \tilde{\mathbf{S}} \cdot \tilde{\mathbf{F}}^T$ and $\mathrm{dev}\left[\cdot\right]$ is the deviatoric operator in the spatial frame.

For these materials, the entire bulk (volumetric) behavior is determined by the function $U\left(J\right)$, and p represents the entire hydrostatic stress. The function $U\left(J\right)$ is constructed to have a value of 0 for J =1 and to have a positive value for all other values of J >0.

All of these materials make use of the three-field element described by Simo and Taylor [73]. This element uses a trilinear interpolation of the displacement field and piecewise-constant interpolations for the pressure and volume ratio.

The uncoupled materials and the associated three-field element are very effective for representing nearly incompressible material behavior. Fully incompressible behavior can be obtained using an augmented Lagrangian method. To use the augmented Lagrangian method for enforcement of the incompressibility constraint to a user-defined tolerance, the user must define two additional material parameters:

Parameter	Description	Default
<laugon>	Turn augmented Lagrangian on for this material or off (1)	0 (off)
<atol>	Augmentation tolerance (2)	0.01

Comments:

1. A value of *1* (one) turns augmentation on, where a value of 0 (zero) turns it off.

2. The augmentation tolerance determines the convergence condition that is used for the augmented Lagrangian method: convergence is reached when the relative ratio of the lagrange multiplier norm of the previous augmentation $\|\lambda_k\|$ to the current one $\|\lambda_{k+1}\|$ is less than the specified value:

$$\left| \frac{\|\lambda_{k+1}\| - \|\lambda_k\|}{\|\lambda_{k+1}\|} \right| < \varepsilon.$$

Thus, a value of 0.01 implies that the change in norm between the previous augmentation loop and the current loop is less than 1%.

The augmented Lagrangian method for incompressibility enforcement is available for all materials that are based on an uncoupled hyperelastic strain energy function.

Example:

```
<material id="1" type="Mooney-Rivlin">
  <c1>5</c1>
  <c2>0.4</c2>
  <k>10000</k>
  <laugon>1</laugon> ← turns on augmented Lagrangian iterations
  <atol>0.05</atol> ← sets the augmentation tolerance
</material>
```

4.1.2.1 Arruda-Boyce

This material describes an incompressible Arruda-Boyce model [3]. The following material parameters are required:

<mu>	initial modulus	[**P**]
<N>	number of links in chain	[ ]
<k>	Bulk modulus	[**P**]

The uncoupled strain energy function for the Arruda-Boyce material is given by:

$$\Psi = \mu \sum_{i=1}^{5} \frac{C_i}{N^{i-1}} \left(\tilde{I}_1^i - 3^i \right) + U(J),$$

where, $C_1 = \frac{1}{2}, C_2 = \frac{1}{20}, C_3 = \frac{11}{1050}, C_4 = \frac{19}{7000}, C_5 = \frac{519}{673750}$ and $I_1$the first invariant of the right Cauchy-Green tensor. The volumetric strain function U is defined as follows,

$$U(J) = \frac{1}{2} k (\ln J)^2 .$$

This material model was proposed by Arruda and Boyce [3] and is based on an eight-chain representation of the macromolecular network structure of polymer chains. The strain energy form represents a truncated Taylor series of the inverse Langevin function, which arises in the statistical treatment of non-Gaussian chains. The parameter N is related to the locking stretch λ_L, the stretch at which the chains reach their full extended state, by $\lambda_L = \sqrt{N}$.

Example:

```
<material id="1" type="Arruda-Boyce">
  <mu>0.09</mu>
  <N>26.5</N>
  <k>100</k>
</material>
```

4.1.2.2 Ellipsoidal Fiber Distribution Uncoupled

The material type for an ellipsoidal continuous fiber distribution in an uncoupled formulation is "*EFD uncoupled*". Since fibers can only sustain tension, this material is not stable on its own. It must be combined with a stable uncoupled material that acts as a ground matrix, using a "*uncoupled solid mixture*" container as described in Section 4.1.2.16. The following material parameters need to be defined:

<beta>	parameters $(\beta_1, \beta_2, \beta_3)$	[ ]
<ksi>	parameters (ξ_1, ξ_2, ξ_3)	[**P**]

The stress $\tilde{\boldsymbol{\sigma}}$ for this material is given by [52, 4, 6]:

$$\tilde{\boldsymbol{\sigma}} = \int_0^{2\pi} \int_0^{\pi} H\left(\tilde{I}_n - 1\right) \tilde{\sigma}_n\left(\mathbf{n}\right) \sin\varphi \, d\varphi \, d\theta.$$

$\tilde{I}_n = \tilde{\lambda}_n^2 = \mathbf{N} \cdot \tilde{\mathbf{C}} \cdot \mathbf{N}$ is the square of the fiber stretch $\tilde{\lambda}_n$, $\mathbf{N}$ is the unit vector along the fiber direction (in the reference configuration), which in spherical angles is directed along (θ, φ), $\mathbf{n} = \tilde{\mathbf{F}} \cdot \mathbf{N}/\tilde{\lambda}_n$, and $H\left(.\right)$ is the unit step function that enforces the tension-only contribution. The fiber stress is determined from a fiber strain energy function in the usual manner,

$$\tilde{\sigma}_n = \frac{2\tilde{I}_n}{J} \frac{\partial \tilde{\Psi}}{\partial \tilde{I}_n} \mathbf{n} \otimes \mathbf{n} \,,$$

where in this material,

$$\tilde{\Psi}\left(\mathbf{n}, \tilde{I}_n\right) = \xi\left(\mathbf{n}\right) \left(\tilde{I}_n - 1\right)^{\beta(\mathbf{n})} \,.$$

The materials parameters β and ξ are determined from:

$$\xi\left(\mathbf{n}\right) = \left(\frac{\cos^2\theta \sin^2\varphi}{\xi_1^2} + \frac{\sin^2\theta \sin^2\varphi}{\xi_2^2} + \frac{\cos^2\varphi}{\xi_3^2} \right)^{-1/2}$$
$$\beta\left(\mathbf{n}\right) = \left(\frac{\cos^2\theta \sin^2\varphi}{\beta_1^2} + \frac{\sin^2\theta \sin^2\varphi}{\beta_2^2} + \frac{\cos^2\varphi}{\beta_3^2} \right)^{-1/2} .$$

The orientation of the material axis can be defined as explained in detail in Section 4.1.1.

Example:

```
<material id="1" type="uncoupled solid mixture">
  <mat_axis type="local">0,0,0</mat_axis>
  <k>1000</k>
  <solid type="Mooney-Rivlin">
    <c1>1</c1>
    <c2>0</c2>
  </solid>
  <solid type="EFD uncoupled">
    <ksi>10, 12, 15</ksi>
    <beta>2.5, 3, 3</beta>
  </solid>
</material>
```

4.1.2.3 Ellipsoidal Fiber Distribution Mooney-Rivlin

The material type for a Mooney-Rivlin material with an ellipsoidal continuous fiber distribution is "*EFD Mooney-Rivlin*". The following material parameters need to be defined:

<c1>	Mooney-Rivlin parameter c1	[**P**]
<c2>	Mooney-Rivlin parameter c2	[**P**]
<k>	bulk modulus	[**P**]
<beta>	parameters $(\beta_1, \beta_2, \beta_3)$	[ ]
<ksi>	parameters (ξ_1, ξ_2, ξ_3)	[**P**]

The stress $\tilde{\boldsymbol{\sigma}}$ for this material is given by,

$$\tilde{\boldsymbol{\sigma}} = \tilde{\boldsymbol{\sigma}}_{MR} + \tilde{\boldsymbol{\sigma}}_f \, .$$

Here, $\tilde{\boldsymbol{\sigma}}_{MR}$ is the stress from the Mooney-Rivlin basis (Section 4.1.2.9), and $\tilde{\boldsymbol{\sigma}}_f$ is the stress contribution from the ellipsoidal fiber distribution (Section 4.1.2.2). The orientation of the material axes can be defined as explained in detail in Section 4.1.1.

Example:

```
<material id="1" type="EFD Mooney-Rivlin">
  <c1>1</c1>
  <c2>0</c2>
  <beta>4.5,4.5,4.5</beta>
  <ksi>1,1,1</ksi>
  <k>20000</k>
  <mat_axis type="local">0,0,0</mat_axis>
</material>
```

4.1.2.4 Ellipsoidal Fiber Distribution Veronda-Westmann

The material type for a Veronda-Westmann material with an ellipsoidal continuous fiber distribution is "*EFD Veronda-Westmann*". The following material parameters need to be defined:

<c1>	First VW coefficient	[**P**]
<c2>	Second VW coefficient	[ ]
<k>	Bulk modulus	[**P**]
<beta>	parameters $(\beta_1, \beta_2, \beta_3)$	[ ]
<ksi>	parameters (ξ_1, ξ_2, ξ_3)	[**P**]

The stress $\tilde{\boldsymbol{\sigma}}$ for this material is given by,

$$\tilde{\boldsymbol{\sigma}} = \tilde{\boldsymbol{\sigma}}_{VW} + \tilde{\boldsymbol{\sigma}}_f \,.$$

Here, $\tilde{\boldsymbol{\sigma}}_{VW}$ is the stress from the Veronda-Westmann basis (Section 4.1.2.17), and $\tilde{\boldsymbol{\sigma}}_f$ is the stress contribution from the ellipsoidal fiber distribution (Section 4.1.2.2).

Example:

```
<material id="1" type="EFD Veronda-Westmann">
  <c1>1</c1>
  <c2>0.5</c2>
  <k>1000</k>
  <beta>4.5,4.5,4.5</beta>
  <ksi>1,1,1</ksi>
  <mat_axis type="local">0,0,0</mat_axis>
</material>
```

4.1.2.5 Fung Orthotropic

The material type for orthotropic Fung elasticity [33, 32] is "*Fung orthotropic*". The following material parameters must be defined:

<E1>	E_1 Young's modulus	**[P]**
<E2>	E_2 Young's modulus	**[P]**
<E3>	E_3 Young's modulus	**[P]**
<G12>	G_{12} shear modulus	**[P]**
<G23>	G_{23} shear modulus	**[P]**
<G31>	G_{31} shear modulus	**[P]**
<v12>	ν_{12} Poisson's ratio	[ ]
<v23>	ν_{23} Poisson's ratio	[ ]
<v31>	ν_{31} Poisson's ratio	[ ]
<c>	c coefficient	**[P]**
<k>	bulk modulus	**[P]**

The hyperelastic strain energy function is given by [5],

$$\Psi = \frac{1}{2}c\left(e^{\tilde{Q}} - 1\right) + U\left(J\right) , \tag{4.1.6}$$

where,

$$\tilde{Q} = c^{-1}\sum_{a=1}^{3}\left[2\mu_a\mathbf{M}_a : \tilde{\mathbf{E}}^2 + \sum_{b=1}^{3}\lambda_{ab}\left(\mathbf{M}_a : \tilde{\mathbf{E}}\right)\left(\mathbf{M}_b : \tilde{\mathbf{E}}\right)\right] .$$

Here, $\tilde{\mathbf{E}} = \left(\tilde{\mathbf{C}} - \mathbf{I}\right)/2$ and $\mathbf{M}_a = \mathbf{V}_a \otimes \mathbf{V}_a$ where $\mathbf{V}_a$ defines the initial direction of material axis a. See Section 4.1.1.2 on how to define the material axes for orthotropic materials. The Lamé constants μ_a $(a = 1, 2, 3)$ and λ_{ab} $(a, b = 1, 2, 3,\ \lambda_{ba} = \lambda_{ab})$ are related to Young's moduli E_a, shear moduli G_{ab} and Poisson's ratios ν_{ab} via

$$\begin{bmatrix} \lambda_{11}+2\mu_1 & \lambda_{12} & \lambda_{13} & 0 & 0 & 0 \\ \lambda_{12} & \lambda_{22}+2\mu_2 & \lambda_{23} & 0 & 0 & 0 \\ \lambda_{13} & \lambda_{23} & \lambda_{33}+2\mu_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2}\left(\mu_1+\mu_2\right) & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2}\left(\mu_2+\mu_3\right) & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2}\left(\mu_1+\mu_3\right) \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} \frac{1}{E_1} & -\frac{\nu_{12}}{E_1} & -\frac{\nu_{13}}{E_1} & 0 & 0 & 0 \\ -\frac{\nu_{21}}{E_2} & \frac{1}{E_2} & -\frac{\nu_{23}}{E_2} & 0 & 0 & 0 \\ -\frac{\nu_{31}}{E_3} & -\frac{\nu_{32}}{E_3} & \frac{1}{E_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{12}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{23}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{31}} \end{bmatrix} .$$

The orthotropic Lamé parameters should produce a positive definite stiffness matrix.

Example:

```
<material id="3" type="Fung orthotropic">
```

```
    <E1>124</E1>
    <E2>124</E2>
    <E3>36</E3>
    <G12>67</G12>
    <G23>40</G23>
    <G31>40</G31>
    <v12>-0.075</v12>
    <v23>0.87</v23>
    <v31>0.26</v31>
    <c>1</c>
    <k>120000</k>
  </material>
```

4.1.2.6 Gent Material

The uncoupled Gent material is defined using the *"Gent"* type string. It defines the following parameters.

<G>	Shear modulus	[**P**]
<Jm>	$J_m = I_m - 1$, with I_m max value for first invariant I_1	[**P**]
<k>	Bulk modulus	[**P**]

The strain energy of this material is defined via an uncoupled formulation,

$$\Psi_r = \tilde{\Psi}_r \left(\tilde{\mathbf{C}} \right) + U\left(J\right)$$

Here, the deviatoric strain energy is given by,

$$\tilde{\Psi} = -\frac{\mu J_m}{2} \ln \left(1 - \frac{\tilde{I}_1 - 3}{J_m} \right)$$

Example:

```
<material id="2" type="Gent">
  <G>3.14</G>
  <Jm>1.5</Jm>
  <k>1e5</k>
</material>
```

4.1.2.7 Uncoupled Holmes-Mow

The material type for the uncoupled Holmes-Mow material is *uncoupled Holmes-Mow*. The following material parameters must be defined:

<mu>	Shear modulus μ	[**P**]
<beta>	Exponential stiffening coefficient β	[]

This material model uncouples deviatoric and volumetric behaviors,

$$\Psi_r = \tilde{\Psi}_r \left(\tilde{\mathbf{C}} \right) + U\left(J\right)$$

The deviatoric strain-energy function is given by

$$\tilde{\Psi}_r = \frac{1}{2} \frac{\mu}{\beta} \left(e^{\tilde{Q}} - 1 \right)$$

where μ is the shear modulus and

$$\tilde{Q} = \beta \left(\tilde{I}_1 - 3 \right)$$

where β is the exponential stiffening coefficient.

Example:

```
<material id="1" name="Material1" type="uncoupled Holmes-Mow">
  <density>1</density>
  <mu>0.5</mu>
  <beta>2</beta>
  <k>5000</k>
</material>
```

4.1.2.8 Holzapfel-Gasser-Ogden

The material type for the uncoupled Holzapfel-Gasser-Ogden material [35] is *Holzapfel-Gasser-Ogden*. The following material parameters must be defined:

<c>	Shear modulus of ground matrix	[**P**]
<k1>	Fiber modulus	[**P**]
<k2>	Fiber exponential coefficient	[**P**]
<gamma>	Fiber mean orientation angle γ	[**deg**]
<kappa>	Fiber dispersion	[]
<k>	Bulk modulus	[**P**]

This material model uncouples deviatoric and volumetric behaviors. The deviatoric strain-energy function is given by:

$$\Psi_r = \tilde{\Psi}_r\left(\tilde{\mathbf{C}}\right) + U\left(J\right)$$

where

$$\tilde{\Psi}_r = \frac{c}{2}\left(\tilde{I}_1 - 3\right) + \frac{k_1}{2k_2}\sum_{\alpha=1}^{2}\left(\exp\left(k_2\left\langle\tilde{E}_\alpha\right\rangle^2\right) - 1\right)$$

and the default volumetric strain energy function is

$$U\left(J\right) = \frac{k}{2}\left(\frac{J^2-1}{2} - \ln J\right)$$

The fiber strain is

$$\tilde{E}_\alpha = \kappa\left(\tilde{I}_1 - 3\right) + \left(1 - 3\kappa\right)\left(\tilde{I}_{4\alpha} - 1\right)$$

where $\tilde{I}_1 = \operatorname{tr}\tilde{\mathbf{C}}$ and $\tilde{I}_{4\alpha} = \mathbf{a}_{\alpha r}\cdot\tilde{\mathbf{C}}\cdot\mathbf{a}_{\alpha r}$. The Macaulay brackets around $\left\langle\tilde{E}_\alpha\right\rangle$ indicate that this term is zero when $\tilde{E}_\alpha < 0$ and equal to $\tilde{E}_\alpha$ when this strain is positive.

There are two fiber families along the vectors $\mathbf{a}_{\alpha r}$ $(\alpha = 1, 2)$, lying in the $\{\mathbf{e}_1, \mathbf{e}_2\}$ plane of the local material axes $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$, making an angle $\pm\gamma$ with respect to $\mathbf{e}_1$. Each fiber family has a dispersion κ, where $0 \leq \kappa \leq \frac{1}{3}$. When $\kappa = 0$ there is no fiber dispersion, implying that all the fibers in that family act along the angle $\pm\gamma$; the value $\kappa = \frac{1}{3}$ represents an isotropic fiber dispersion. All other intermediate values of κ produce a $\pi-$periodic von Mises fiber distribution, as described in [35]. c is the shear modulus of the ground matrix; k_1 is the fiber modulus and k_2 is the exponential coefficient.

Example:

```
<material id="2" type="Holzapfel-Gasser-Ogden">
  <c>7.64</c>
  <k1>996.6</k1>
  <k2>524.6</k2>
  <gamma>49.98</gamma>
  <kappa>0.226</kappa>
  <k>1e5</k>
</material>
```

4.1.2.9 Mooney-Rivlin

The material type for uncoupled Mooney-Rivlin materials is *Mooney-Rivlin*. The following material parameters must be defined:

<c1>	Coefficient of first invariant term	[**P**]
<c2>	Coefficient of second invariant term	[**P**]
<k>	Bulk modulus	[**P**]

This material model is a hyperelastic Mooney-Rivlin type with uncoupled deviatoric and volumetric behavior. The strain-energy function is given by:

$$\Psi = C_1 \left(\tilde{I}_1 - 3 \right) + C_2 \left(\tilde{I}_2 - 3 \right) + \frac{1}{2} K \left(\ln J \right)^2 .$$

C_1 and C_2 are the Mooney-Rivlin material coefficients. The variables $\tilde{I}_1$ and $\tilde{I}_2$ are the first and second invariants of the deviatoric right Cauchy-Green deformation tensor $\tilde{\mathbf{C}}$. The coefficient K is a bulk modulus-like penalty parameter and J is the determinant of the deformation gradient tensor. When $C_2 = 0$, this model reduces to an uncoupled version of the neo-Hookean constitutive model.

This material model uses a three-field element formulation, interpolating displacements as linear field variables and pressure and volume ratio as piecewise constant on each element [73].

This material model is useful for modeling certain types of isotropic materials that exhibit some limited compressibility, i.e. 100 < (K/C_1) < 10000.

Example:

```
<material id="2" type="Mooney-Rivlin">
  <c1>10.0</c1>
  <c2>20.0</c2>
  <k>1000</k>
</material>
```

4.1.2.10 Muscle Material

This material model implements the constitutive model developed by Silvia S. Blemker [21]. The material type for the muscle material is *muscle material*. The model is designed to simulate the passive and active material behavior of skeletal muscle. It defines the following parameters:

<g1>	along fiber shear modulus	[**P**]
<g2>	cross fiber shear modulus	[**P**]
<p1>	exponential stress coefficients	[ ]
<p2>	fiber uncrimping factor	[ ]
<Lofl>	optimal fiber length	[ ]
<smax>	maximum isometric stress	[**P**]
<lambda>	fiber stretch for straightened fibers	[ ]
<k>	bulk modulus	[**P**]
<activation>	activation level	

The main difference between this material formulation compared to other transversely hyperelastic materials is that it is formulated using a set of new invariants, originally due to Criscione [26], instead of the usual five invariants proposed by A.J.M. Spencer [76]. For this particular material, only two of the five Criscione invariants are used. The strain energy function is defined as follows:

$$\Psi\left(B_1, B_2, \lambda\right) = G_1 \tilde{B}_1^2 + G_2 \tilde{B}_2^2 + F_m\left(\tilde{\lambda}\right) + U\left(J\right).$$

The function $F_m$is the strain energy contribution of the muscle fibers. It is defined as follows:

$$\lambda \frac{\partial F_m}{\partial \lambda} = \sigma_{\max}\left(f_m^{\text{passive}}\left(\lambda\right) + \alpha f_m^{\text{active}}\left(\lambda\right)\right)\frac{\lambda}{\lambda_{ofl}},$$

where,

$$f_m^{\text{passive}}\left(\lambda\right) = \begin{cases} 0 & \lambda \leqslant \lambda_{ofl} \\ P_1\left(e^{P_2\left(\lambda/\lambda_{ofl}-1\right)} - 1\right) & \lambda_{ofl} < \lambda < \lambda^* \\ P_3\lambda/\lambda_{ofl} + P_4 & \lambda \geqslant \lambda^* \end{cases},$$

and

$$f_m^{\text{active}}\left(\lambda\right) = \begin{cases} 9\left(\lambda/\lambda_{ofl} - 0.4\right)^2 & \lambda \leqslant 0.6\lambda_{ofl} \\ 9\left(\lambda/\lambda_{ofl} - 1.6\right)^2 & \lambda \geqslant 1.4\lambda_{ofl} \\ 1 - 4\left(1 - \lambda/\lambda_{ofl}\right)^2 & 0.6\lambda_{ofl} < \lambda < 1.4\lambda_{ofl} \end{cases}.$$

The values $P_3$and P_4 are determined by requiring $C^0$and C^1continuity at $\lambda = \lambda^*$.

The parameter α is the activation level and can be specified using the *activation* element. You can specify a loadcurve using the *lc* attribute. The value is interpreted as a scale factor when a loadcurve is defined or as the constant activation level when no loadcurve is defined.

The muscle fiber direction is specified similarly to the transversely isotropic Mooney-Rivlin model.

Example:

```
<material id="1" type="muscle material">
  <g1>500</g1>
  <g2>500</g2>
```

```
    <p1>0.05</p1>
    <p2>6.6</p2>
    <smax>3e5</smax>
    <Lofl>1.07</Lofl>
    <lambda>1.4</lambda>
    <k>1e6</k>
    <fiber type="vector">1,0,0</fiber>
  </material>
```

4.1.2.11 Ogden

This material describes an incompressible hyperelastic Ogden material [73]. The following material parameters must be defined:

<c[n]>	Coefficient of n^{th} term, where n can range from 1 to 6	[**P**]
<m[n]>	Exponent of n^{th} term, where n can range from 1 to 6	[ ]
<k>	Bulk modulus	[**P**]

The uncoupled hyperelastic strain energy function for this material is given in terms of the eigenvalues of the deformation tensor:

$$\Psi = \sum_{i=1}^{N} \frac{c_i}{m_i^2} \left(\tilde{\lambda}_1^{m_i} + \tilde{\lambda}_2^{m_i} + \tilde{\lambda}_3^{m_i} - 3 \right) + U\left(J\right).$$

Here, $\tilde{\lambda}_i^2$ are the eigenvalues of $\tilde{\mathbf{C}}$, $c_i$and $m_i$are material coefficients and N ranges from 1 to 6. Note that you only have to include the material parameters for the terms you intend to use.

Example:

```
<material id="1" type="Ogden">
  <m1>2.4</m1>
  <c1>1</c1>
  <k>100</k>
</material>
```

4.1.2.12 Tendon Material

The material type for the tendon material is *tendon material*. The tendon material is similar to the muscle material [21]. The only difference is the fiber function. For tendon material this is defined as:

$$\lambda \frac{\partial F_t}{\partial \lambda} = \sigma(\lambda),$$

where

$$\sigma(\lambda) = \begin{cases} 0 & \lambda \leqslant 1 \\ L_1\left(e^{L_2(\lambda-1)} - 1\right) & 1 < \lambda < \lambda^* \\ L_3\lambda + L_4 & \lambda \geqslant \lambda^* \end{cases}.$$

The parameters $L_3$and $L_4$are determined by requiring C^0 and C^1 continuity at λ^*.

The material parameters for this material are listed below.

<g1>	along fiber shear modulus	[**P**]
<g2>	cross fiber shear modulus	[**P**]
<l1>	exponential stress coefficients	[**P**]
<l2>	fiber uncrimping factor	[ ]
<lambda>	fiber stretch for straightened fibers	[ ]
<k>	bulk modulus	[**P**]

The tendon fiber direction is specified similarly to the transversely isotropic Mooney-Rivlin model.

Example:

```
<material id="1" type="tendon material">
  <g1>5e4</g1>
  <g2>5e4</g2>
  <l1>2.7e6/l1>
  <l2>46.4</l2>
  <lambda>1.03</lambda>
  <k>1e7</k>
  <fiber type="vector">1,0,0</fiber>
</material>
```

4.1.2.13 Tension-Compression Nonlinear Orthotropic

The material type for the tension-compression nonlinear orthotropic material is "*TC nonlinear orthotropic*". The following material parameters are defined:

<c1>	First Mooney-Rivlin material parameter	[**P**]
<c2>	Second Mooney-Rivlin material parameter	[**P**]
<k>	bulk modulus	[**P**]
<beta>	the β parameter (see below)	[ ]
<ksi>	the ξ parameter (see below)	[**P**]
<mat_axis>	defines the material axes	

This material is based on the following uncoupled hyperelastic strain energy function [13]:

$$\Psi\left(\mathbf{C},\lambda_1,\lambda_2,\lambda_3\right)=\tilde{\Psi}_{iso}\left(\tilde{\mathbf{C}}\right)+\sum_{i=1}^{3}\tilde{\Psi}_i^{TC}\left(\tilde{\lambda}_i\right)+U\left(J\right)\ .$$

The isotropic strain energy $\tilde{\Psi}_{iso}$and the dilatational energy U are the same as for the Mooney-Rivlin material. The tension-compression term is defined as follows:

$$\tilde{\Psi}_i^{TC}\left(\tilde{\lambda}_i\right)=\begin{cases}\xi_i\left(\tilde{\lambda}_i-1\right)^{\beta_i} & \tilde{\lambda}_i>1\\ 0 & \tilde{\lambda}_i\leqslant 1\end{cases}\qquad \xi_i\geqslant 0\quad \text{(no sum over } i\text{)}\ .$$

The $\tilde{\lambda}_i$ parameters are the deviatoric fiber stretches of the local material fibers:

$$\tilde{\lambda}_i=\left(\mathbf{a}_i^0\cdot\tilde{\mathbf{C}}\cdot\mathbf{a}_i^0\right)^{1/2}\ .$$

The local material fibers are defined (in the reference frame) as an orthonormal set of vectors $\mathbf{a}_i^0$. See Section 4.1.1 for more information. As with all uncoupled materials, this material uses the three-field element formulation.

A complete example for this material follows.

```
<material id="7" name="cartilage" type="TC nonlinear orthotropic">
  <c1>1.0</c1>
  <c2>0.0</c2>
  <k>100</k>
  <beta>4.3,4.3,4.3</beta>
  <ksi>4525, 4525, 4525</ksi>
  <mat_axis type="local">0,0,0</mat_axis>
</material>
```

4.1.2.14 Transversely Isotropic Mooney-Rivlin

The material type for transversely isotropic Mooney-Rivlin materials is "*trans iso Mooney-Rivlin*". The following material parameters must be defined:

<c1>	Mooney-Rivlin coefficient 1	[**P**]
<c2>	Mooney-Rivlin coefficient 2	[**P**]
<c3>	Exponential stress coefficient	[**P**]
<c4>	Fiber uncrimping coefficient	[ ]
<c5>	Modulus of straightened fibers	[**P**]
<k>	Bulk modulus	[**P**]
<lam_max>	Fiber stretch for straightened fibers	[ ]
<fiber>	Fiber distribution option	
<active_contraction>	Optional active contraction	

This constitutive model can be used to represent a material that has a single preferred fiber direction and was developed for application to biological soft tissues [65, 66, 83]. It can be used to model tissues such as tendons, ligaments and muscle. The elastic response of the tissue is assumed to arise from the resistance of the fiber family and an isotropic matrix. It is assumed that the uncoupled strain energy function can be written as follows:

$$\Psi = F_1\left(\tilde{I}_1, \tilde{I}_2\right) + F_2\left(\tilde{\lambda}\right) + \frac{K}{2}\left[\ln\left(J\right)\right]^2.$$

Here $\tilde{I}_1$ and $\tilde{I}_2$ are the first and second invariants of the deviatoric version of the right Cauchy Green deformation tensor $\tilde{\mathbf{C}}$ and $\tilde{\lambda}$ is the deviatoric part of the stretch along the fiber direction ($\tilde{\lambda}^2 = \mathbf{a}_0 \cdot \tilde{\mathbf{C}} \cdot \mathbf{a}_0$, where $\mathbf{a}_0$ is the initial fiber direction), and $J = \det\left(\mathbf{F}\right)$ is the Jacobian of the deformation (volume ratio). The function F_1 represents the material response of the isotropic ground substance matrix and is the same as the Mooney-Rivlin form specified above, while F_2 represents the contribution from the fiber family. The strain energy of the fiber family is as follows:

$$F_2\left(\tilde{\lambda}\right) = \begin{cases} 0 & \tilde{\lambda} \leqslant 1 \\ C_3\left(e^{-C_4}\left(\mathrm{Ei}\left(C_4\tilde{\lambda}\right) - \mathrm{Ei}\left(C_4\right)\right) - \ln\tilde{\lambda}\right) & 1 < \tilde{\lambda} < \lambda_m \\ C_5\left(\tilde{\lambda} - 1\right) + C_6\ln\tilde{\lambda} & \tilde{\lambda} \geqslant \lambda_m \end{cases}$$

where $\mathrm{Ei}\left(\cdot\right)$ is the exponential integral function. The resulting fiber stress is evaluated from

$$\tilde{\lambda}\frac{\partial F_2}{\partial\tilde{\lambda}} = \begin{cases} 0 & \tilde{\lambda} \leqslant 1 \\ C_3\left(e^{C_4\left(\tilde{\lambda}-1\right)} - 1\right) & 1 < \tilde{\lambda} < \lambda_m \\ C_5\tilde{\lambda} + C_6 & \tilde{\lambda} \geqslant \lambda_m \end{cases}.$$

Here, C_1 and C_2 are the Mooney-Rivlin material coefficients, *lam_max* (λ_m) is the stretch at which the fibers are straightened, C_3scales the exponential stresses, C_4 is the rate of uncrimping of the fibers, and $C_5$is the modulus of the straightened fibers. C_6 is determined from the requirement that the stress is continuous at λ_m.

This material model uses a three-field element formulation, interpolating displacements as linear field variables and pressure and volume ratio as piecewise constant on each element [73].

The fiber orientation can be specified as explained in Section 4.1.1. An optional active contraction model can also be defined for this material. See Section 4.1.3 for more details.

Example:

This example defines a transversely isotropic material with a Mooney-Rivlin basis. It defines a homogeneous fiber direction and uses the active fiber contraction feature.

```
<material id="3" type="trans iso Mooney-Rivlin">
  <c1>13.85</c1>
  <c2>0.0</c2>
  <c3>2.07</c3>
  <c4>61.44</c4>
  <c5>640.7</c5>
  <k>100.0</k>
  <lam_max>1.03</lam_max>
  <fiber type="vector">1,0,0</fiber>
</material>
```

4.1.2.15 Transversely Isotropic Veronda-Westmann

The material type for transversely isotropic Veronda-Westmann materials is "*trans iso Veronda-Westmann*" [82]. The following material parameters must be defined:

<c1>	Veronda-Westmann coefficient 1	[**P**]
<c2>	Veronda-Westmann coefficient 2	[ ]
<c3>	Exponential stress coefficient	[**P**]
<c4>	Fiber uncrimping coefficient	[ ]
<c5>	Modulus of straightened fibers	[**P**]
<k>	Bulk modulus	[**P**]
<lam_max>	Fiber stretch for straightened fibers	[ ]
<fiber>	Fiber distribution option.	
<active_contraction>	Optional active contraction	

This uncoupled hyperelastic material differs from the Transversely Isotropic Mooney-Rivlin model in that it uses the Veronda-Westmann model for the isotropic matrix. The interpretation of the material parameters, except C_1 and $C_2$is identical to this material model.

The fiber distribution option is explained in Section 4.1.1. An optional active contraction model can also be defined for this material. See Section 4.1.3 for more details.

Example:

This example defines a transversely isotropic material model with a Veronda-Westmann basis. The fiber direction is implicitly implied as *local*.

```
<material id="3" type="trans iso Veronda-Westmann">
  <c1>13.85</c1>
  <c2>0.0</c2>
  <c3>2.07</c3>
  <c4>61.44</c4>
  <c5>640.7</c5>
  <lam_max>1.03</lam_max>
</material>
```

4.1.2.16 Uncoupled Solid Mixture

The material type for for a constrained mixture of uncoupled solids is "*uncoupled solid mixture*". This material describes a mixture of quasi-incompressible elastic solids. It is a container for any combination of the materials described in Section 4.1.2.

<solid>	Container tag for solid material	

The mixture may consist of any number of solids. The stress tensor for the solid mixture is the sum of the stresses for all the solids. The bulk modulus of the uncoupled solid mixture is the sum of the bulk moduli of the individual <solid> materials. A bulk modulus specified outside of the <solid> materials will be ignored.

Material axes may be optionally specified within the <material> level, as well as within each <solid>. Within the <material> level, these represent the local element axes relative to the global coordinate system. Within the <solid>, they represent local material axes relative to the element. If material axes are specified at both levels, they are properly compounded to produce local material axes relative to the global coordinate system. Material axes specified in the <ElementData> section are equivalent to a specification at the <material> level: they correspond to local element axes relative to the global system.

Example:

```
<material id="1" type="uncoupled solid mixture">
  <k>30e3</k>
  <mat_axis type="vector">
    <a>1,0,0</a>
    <d>0,1,0</d>
  </mat_axis>
  <solid type="Mooney-Rivlin">
    <c1>2.0</c1>
    <c2>0.0</c2>
  </solid>
  <solid type="EFD uncoupled">
    <mat_axis type="vector">
      <a>0.8660254,0.5,0</a>
      <d>0,0,1</d>
    </mat_axis>
    <ksi>5, 1, 1</ksi>
    <beta>2.5, 3, 3</beta>
  </solid>
  <solid type="EFD uncoupled">
    <mat_axis type="vector">
      <a>0.8660254,-0.5,0</a>
      <d>0,0,1</d>
    </mat_axis>
    <ksi>5, 1, 1</ksi>
    <beta>2.5, 3, 3</beta>
  </solid>
</material>
```

4.1.2.17 Veronda-Westmann

The material type for incompressible Veronda-Westmann materials is *Veronda-Westmann* [78]. The following material parameters must be defined:

<c1>	First VW coefficient	[**P**]
<c2>	Second VW coefficient	[ ]
<k>	Bulk modulus	[**P**]

This model is similar to the Mooney-Rivlin model in that it also uses an uncoupled deviatoric dilatational strain energy:

$$\Psi = C_1 \left[e^{\left(C_2\left(\tilde{I}_1 - 3\right)\right)} - 1 \right] - \frac{C_1 C_2}{2} \left(\tilde{I}_2 - 3 \right) + U\left(J\right) .$$

The dilatational term is identical to the one used in the Mooney-Rivlin model. This model can be used to describe certain types of biological materials that display exponential stiffening with increasing strain. It has been used to describe the response of skin tissue [78].

Example:

```
<material id="2" type="Veronda-Westmann">
  <c1>1000.0</c1>
  <c2>2000.0</c2>
  <k>1000</k>
</material>
```

4.1.2.18 Mooney-Rivlin Von Mises Distributed Fibers

(Sclera and other thin soft tissues)

Authors: Cécile L.M. Gouget and Michaël J.A. Girard

The material type for a thin material where fiber orientation follows a von Mises distribution is "Mooney-Rivlin von Mises Fibers". The following parameters must be defined:

<c1>	Mooney-Rivlin coefficient 1	[**P**]
<c2>	Mooney-Rivlin coefficient 2	[**P**]
<c3>	Exponential stress coefficient	[**P**]
<c4>	Fiber uncrimping coefficient	[ ]
<c5>	Modulus of straightened fibers	[**P**]
<k>	Bulk modulus	[**P**]
<lam_max>	Fiber stretch for straightened fibers	[ ]
<tp>	Preferred fiber orientation in radian (angle within the plane of the fibers, defined with respect to the first direction given in mat_axis)	[rad]
<kf>	Fiber concentration factor	[ ]
<vmc>	Choice of von Mises distribution	
	=1 semi-circular von Mises distribution	
	=2 constrained von Mises mixture distribution	
<var_n>	Exponent (only for vmc=2)	
<gipt>	Number of integration points (value is a multiple of 10)	
<mat_axis>	Reference frame of the plane of the fibers	

This constitutive model is designed for thin soft tissues. Fibers are multi-directional: they are distributed within the plane tangent to the tissue surface and they follow a unimodal distribution. It is a constitutive model that is suitable for ocular tissues (e.g. sclera and cornea) but can be used for other thin soft tissues. The proposed strain energy function is as follows:

$$\Psi = F_1\left(\tilde{I}_1, \tilde{I}_2\right) + \int_{\theta_P-\pi/2}^{\theta_P+\pi/2} P\left(\theta\right) F_2\left(\tilde{\lambda}\left[\theta\right]\right) d\theta + \frac{K}{2}\left[\ln\left(J\right)\right]^2,$$

where P is the 2D unimodal distribution function of the fibers which satisfies the normalization condition:

$$\int_{\theta_P-\pi/2}^{\theta_P+\pi/2} P\left(\theta\right) d\theta = 1.$$

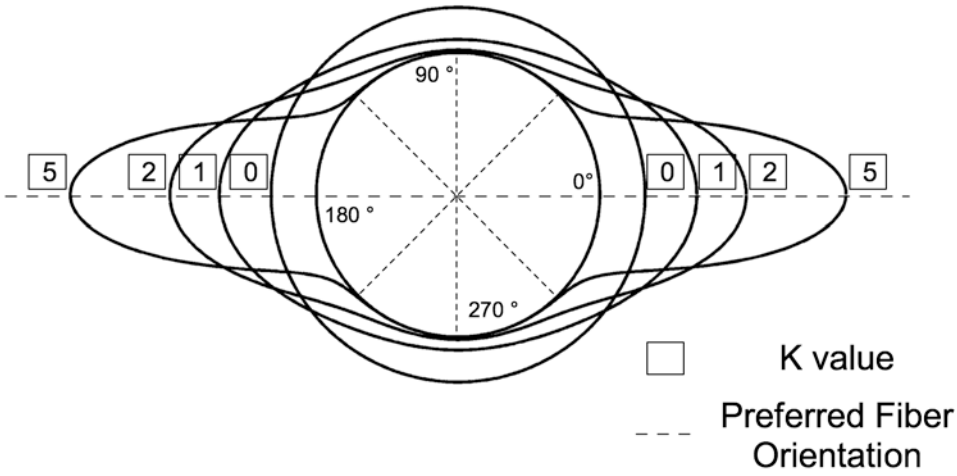
θ_P is the preferred fiber orientation relative to a local coordinate system (parameter tp). The functions $F_1$and F_2 are described in the "Transversely Isotropic Mooney-Rivlin" model. The user can choose between 2 distribution functions using the parameter vmc.

1. Semi-circular von Mises distribution (vmc = 1) [37]

The semi-circular von Mises distribution is one of the simplest unimodal distribution in circular statistics. It can be expressed as

$$P(\theta) = \frac{1}{\pi I_0(k_f)} \exp\left[k_f \cos\left(2(\theta - \theta_p)\right)\right],$$

where I_0 is the modified Bessel function of the first kind (order 0), and k_f is the fiber concentration factor. k_f controls the amount of fibers that are concentrated along the orientation θ_P as illustrated in the figure below.



Polar representation of the semi-circular von-Mises distribution describing in-plane collagen fiber alignment. In this case, the preferred fiber orientation θ_P is equal to zero degrees. When the fiber concentration factor k is equal to zero, the collagen fibers have an isotropic distribution in a plane tangent to the scleral wall. As k increases, the collagen fibers align along the preferred fiber orientation θ_P. Note that the distributions were plotted on a circle of unit one to ease visualization.

2. Constrained von Mises Mixture Distribution (vmc = 2) [38]

The semi-circular von Mises distribution is ideal for its simplicity but in some instance it fails to accurately describe the isotropic subpopulation of fibers present in thin soft tissues. An improved mathematical description is proposed here as a weighted mixture of the semi-circular uniform distribution and the semi-circular von Mises distribution. It can be expressed as:

$$P(\theta) = \frac{1-\beta}{\pi} + \frac{\beta}{\pi I_0(k_f)} \exp\left[k_f \cos\left(2(\theta - \theta_p)\right)\right],$$

where β needs to be constrained for uniqueness and stability. β is expressed as:

$$\beta = \left(\frac{I_1(k_f)}{I_0(k_f)}\right)^n,$$

where I_1 is the modified Bessel function of the first kind (order 1), and n is an exponent to be determined experimentally (parameter var_n). n=2 has been found to be suitable for the sclera based on fiber distribution measurements using small angle light scattering.

Note about Numerical Integration

All numerical integrations are performed using a 10-point Gaussian quadrature rule. The number of Gaussian integration points can be increased for numerical stability. This is controlled through the parameter *gipt*. We recommend to use at least *gipt* = 20 (2×10 integration points).

Note that the more integration points are used, the slower the model will run. By increasing the number of integration points, one should observe convergence in the numerical accuracy. The parameter gipt is required to be a multiple of 10.

Definition of the Fiber Plane

The user must specify two directions to define a local coordinate system of the plane in which the fibers lay. As explained in Section 4.1.1.2, this can be done by defining two directions with the parameter *mat_axis*. The first direction (vector $\mathbf{a}$ of *mat_axis*) must be the reference direction used to define the angle *tp* (θ_P). The second direction (vector $\mathbf{d}$ of mat_axis) can be any other direction in the plane of the fibers. Thus, a fiber at angle θ_P will be along the vector $\mathbf{v} = \cos\theta_P \mathbf{e}_1 + \sin\theta_P \mathbf{e}_2$ where $\mathbf{e}_1 = \mathbf{a}/\|\mathbf{a}\|$, $\mathbf{e}_2 = (\mathbf{e}_3 \times \mathbf{a})/\|\mathbf{e}_3 \times \mathbf{a}\|$, $\mathbf{e}_3 = (\mathbf{a} \times \mathbf{d})/\|\mathbf{a} \times \mathbf{d}\|$, as was defined in Section 4.1.1.2.

Note on parameters kf and tp

The parameters kf and tp can be specified either in the Material section or in the *ElementData* section with the tag MRVonMisesParameters. With this second option the user can define different fiber characteristics for each element separately, which can be useful if fiber orientations or concentrations vary spatially. The parameter mat_axis can also be defined either in the Material section or in the *ElementData* section.

Example

This example defines a thin soft tissue material where fibers are distributed according to a constrained von Mises mixture distribution.

```
<material id="1" name="Material 1" type="Mooney-Rivlin von Mises Fibers">
  <c1>10.0</c1>
  <c2>0.0</c2>
  <c3>50.0</c3>
  <c4>5.0</c4>
  <c5>1</c5>
  <lam_max>10</lam_max>
  <k>1000000.0</k>
  <kf>1</kf>
  <vmc>2</vmc>
  <var_n>2.0</var_n>
  <tp>0.0</tp>
  <gipt>40</gipt>
  <mat_axis type="local">4,1,3</mat_axis>
</material>
```

4.1.2.19 isotropic Lee-Sacks uncoupled

This material implements a Fung-type material, as presented in Kamensy, CMAME 2018. The material formulation is selected by setting the type attribute to "`uncoupled isotropic Lee-Sacks`". It defines the following parameters:

<c0>	shear-modulus of neo-Hookean matrix	[**P**]
<c1>	c1 parameter	[**P**]
<c2>	c2 parameter	[ ]
<tangent_scale>	scale factor for c0 when used in tangent (default = 1)	[ ]
<k>	bulk modulus	[**P**]

The strain-energy density function for this material combines a neo-Hookean matrix with a Fung-type exponential term, and is defined as follows.

$$\Psi = \frac{c_0}{2}\left(I_1 - 3\right) + \frac{c_1}{2}\left(e^{c_2\left(I_1-3\right)^2} - 1\right)$$

As reported in Kamensky, the exponential term may cause convergence difficulties under certain loading conditions. It was reported that increasing the value of $c_0$during the stiffness evaluation may improve convergence. The default value for *tangent_scale* is 1, and thus tangent scaling is not applied. A tangent scale factor of 20 was used in Kamensky.

Example:

```
<material id="1" name="material1" type="uncoupled isotropic Lee-Sacks">
  <k>1000</k>
  <c0>10</c0>
  <c1>0.209</c1>
  <c2>9.046</c2>
</material>
```

4.1.3 Fiber Active Contraction

Some of the material models described in Section 4.1.2 which include a fiber model may provide the option to specify an active contraction along that fiber. This section lists the types of fiber active contraction models.

4.1.3.1 Active Contraction

The default active contraction model is "*active_contraction*". It is based on the formulation of Guccione et al. [39] and reviewed in the *FEBio Theory Manual*. The active stress is evaluated as $T^a\mathbf{a}\otimes\mathbf{a}$, where $\mathbf{a}$ is the unit vector along the fiber in the current configuration, and

$$T^a = T_{\max}\frac{Ca_0^2}{Ca_0^2 + ECa_{50}^2}C(t)$$

where

$$ECa_{50} = \frac{(\mathrm{Ca}_0)_{\max}}{\sqrt{\exp\left[\beta\left(\lambda(t)\, l_r - l_0\right)\right] - 1}}$$

In this expression, $\lambda(t)$ is the fiber stretch ratio at the current time. The activation curve $C(t)$ is represented by the *ascl* property that takes an optional attribute, *lc*, which defines the loadcurve. The list of parameters required for this model is as follows.

<ascl>	Activation scale factor $C(t)$
<ca0>	Intracellular calcium concentration Ca_0 (default=1)
<camax>	Maximum peak intracellular calcium concentration $(\mathrm{Ca}_0)_{\max}$ (default=ca0)
<beta>	tension-sarcomere length relation constant β
<l0>	No tension sarcomere length l_0
<refl>	Unloaded sarcomere length l_r
<Tmax>	Isometric tension under maximal activation at camax $T_{\max}$ (default=1)

Example:

This example defines a transversely isotropic material with a Mooney-Rivlin basis. It defines a homogeneous fiber direction and uses the active fiber contraction feature.

```
<material id="3" type="trans iso Mooney-Rivlin">
  <c1>13.85</c1>
  <c2>0.0</c2>
  <c3>2.07</c3>
  <c4>61.44</c4>
  <c5>640.7</c5>
  <k>100.0</k>
  <lam_max>1.03</lam_max>
  <fiber type="vector">1,0,0</fiber>
  <active_contraction type="active_contraction">
    <ascl lc="1">1</ascl>
    <ca0>4.35</ca0>
    <beta>4.75</beta>
    <l0>1.58</l0>
    <refl>2.04</refl>
  </active_contraction>
</material>
```

4.1.3.2 Force-Velocity Active Contraction

This material model was formulated by Estrada et al. [30] and is called "*force-velocity-Estrada*". It is a modification of the "*active_contraction*" material based on the formulation of Guccione et al. [39] described in Section 4.1.3.1, based on the fading-memory formulation poroposed by Hunter et al. [45]. The active stress is evaluated as $T^a\mathbf{a}\otimes\mathbf{a}$, where $\mathbf{a}$ is the unit vector along the fiber in the current configuration, and

$$T^a = e\left(t\right) \quad \underbrace{T_{\max}\frac{Ca_0^2}{Ca_0^2+ECa_{50}^2}}_{\substack{\text{instantaneous length}\\ \text{dependence}}} \quad \underbrace{\frac{1+aQ\left(t,\lambda\left(t\right)\right)}{1-Q\left(t,\lambda\left(t\right)\right)}}_{\text{force-velocity}}$$

where

$$ECa_{50} = \frac{\left(\text{Ca}_0\right)_{\max}}{\sqrt{\exp\left[\beta\left(\lambda\left(t\right)l_r - l_0\right)\right]-1}}$$

and

$$Q\left(t,\lambda\left(t\right)\right) = \sum_{k=1}^{3} A_k \int_{-\infty}^{t} e^{-\alpha_k(t-\tau)}\dot{\lambda}\left(\tau\right)\,d\tau$$

In these expressions, $\lambda\left(t\right)$ is the fiber stretch ratio at the current time. Here, T^a "is the product of three distinct components: a time-varying normalized activation, $e\left(t\right)$, that defines the time course of force generation throughout the cardiac cycle; an instantaneous length-dependent term that scales the peak possible isometric tension based on the current fiber stretch; and a force-velocity term that dampens the instantaneous force generation based on the rate of shortening of the fibers" [30]. The activation curve $e\left(t\right)$ is represented by the *ascl* property that takes an optional attribute, *lc*, which defines the time-dependent loadcurve. The list of parameters required for this model is as follows.

<ascl>	Activation scale factor $e\left(t\right)$
<ca0>	Intracellular calcium concentration Ca_0 (default=1)
<camax>	Maximum peak intracellular calcium concentration $\left(\text{Ca}_0\right)_{\max}$ (default=ca0)
<beta>	tension-sarcomere length relation constant β
<l0>	No tension sarcomere length l_0
<refl>	Unloaded sarcomere length l_r
<Tmax>	Isometric tension under maximal activation at camax $T_{\max}$ (default=1)
<A1>, <A2>, <A3>	Parameters A_k for $k=1,2,3$
<alpha1>, <alpha2>, <alpha3>	Parameters α_k for $k=1,2,3$
<a_t>	Parameter a
<force_velocity>	Set to true (1) by default. Set to false (0) to turn off the force-velocity term

Example:

This example defines a transversely isotropic material with a Mooney-Rivlin basis. It defines a homogeneous fiber direction and uses the active fiber contraction feature.

```
<material id="3" type="trans iso Mooney-Rivlin">
  <c1>13.85</c1>
  <c2>0.0</c2>
```

```
  <c3>2.07</c3>
  <c4>61.44</c4>
  <c5>640.7</c5>
  <k>100.0</k>
  <lam_max>1.03</lam_max>
  <fiber type="vector">1,0,0</fiber>
  <active_contraction type="force-velocity-Estrada">
    <ascl lc="1">1</ascl>
    <ca0>4.35</ca0>
    <beta>4.75</beta>
    <l0>1.58</l0>
    <refl>1.58</refl>
<Tmax>135.7</Tmax>
<alpha1>30.30303</alpha1>
<A1>50</A1>
<a_t>0.5</a_t>
<force_velocity>1</force_velocity>
  </active_contraction>
</material>
```

4.1.4 Unconstrained Materials

Unlike the materials in Section 4.1.2, these materials do not necessarily assume an additive decomposition of the bulk and deviatoric parts of the strain energy or stress. Further, these materials can only be used with the standard displacement-based finite element formulation, rather than the three-field element formulatoin. They should not be used for nearly-incompressible material behavior due to the potential for element locking.

4.1.4.1 Arruda-Boyce Unconstrained

The material type for an unconstrained Arruda-Boyce material is *Arruda-Boyce unconstrained*. This isotropic Arruda-Boyce [3] 8-chain model has been used in the field of polymer modeling and in cervical biomechanics [69]. It is based on the simple freely-jointed-chain model and can be derived using the Langevin equation of statistical mechanics. The strain energy density for this material takes the form

$$\Psi(I_1, J) = \xi \left(\sqrt{\frac{I_1}{3N}}\beta + \ln \frac{\beta}{\sinh \beta} \right) - \frac{\xi}{3}\sqrt{N}\beta_0 \ln J + \frac{1}{2}\kappa(\ln J^2 - 1 - 2\ln J) \tag{4.1.7}$$

where $I_1 = \operatorname{tr} \mathbf{C}$ is the first invariant of $\mathbf{C}$, $\beta = \mathcal{L}^{-1}\left[\sqrt{\frac{I_1}{3N}}\right]$ is the inverse Langevin equation, and $\beta_0 = \mathcal{L}^{-1}\left[\sqrt{\frac{1}{N}}\right]$. Here, $\xi = nk\Theta$ is the initial fiber modulus with n, k, Θ respectively representing the number of chains per unit volume, Boltzmann's constant, and the absolute temperature. The parameter $N = \zeta^2$ is the number of chain segments, and ζ is the locking stretch that represents the extensibility of the material.

A Taylor series expansion is used to evaluate the inverse Langevin equation. The parameter n_{term} is used to control the number of terms used to evaluate the series; n_{term} must be an integer between 3 and 30.

Langevin mechanics describes a stochastic process with non-Gaussian distribution. It will reduce to a Gaussian distribution when the system approaches equilibrium. Thus, when the material is far from its maximum extensibility, i.e., when $\zeta = \sqrt{N}$ is much larger than the stretch, this model will reduce to a neo-Hookean material.

This material The following parameters must be defined:

<ksi>	Initial fiber modulus ξ	[**P**]
<N>	Number of chain segments N	[ ]
<n_term>	Number of terms in the Taylor series expansion n_{term} $(3 \leq n_{\text{term}} \leq 30)$	[ ]
<kappa>	Bulk modulus κ of ground matrix	[**P**]

Example:

```
<material id="1" name="Soft Tissue" type="Arruda-Boyce unconstrained">
<N>5</N>
<ksi>1</ksi>
<n_term>30</n_term>
<kappa>1</kappa>
</material>
```

4.1.4.2 Carter-Hayes

The material type for a Carter-Hayes material is *Carter-Hayes*. The following parameters must be defined:

<E0>	Young's modulus at reference density E_0	[**P**]
<rho0>	reference density ρ_0	[**M**/**L**^{3}]
<gamma>	exponent of solid-bound molecule density for calculation of Young's modulus γ	[ ]
<v>	Poisson's ratio ν	[ ]
<sbm>	index of solid bound molecule	[ ]

This model describes an unconstrained neo-Hookean material [22] whose Young's modulus is a power-law function of the referential apparent density ρ_r^σ of a solid-bound molecule. It is derived from the following hyperelastic strain-energy function:

$$\Psi_r = \frac{E_Y}{2\left(1+\nu\right)}\left[\frac{1}{2}\left(\operatorname{tr}\mathbf{C}-3\right)-\ln J+\frac{\nu}{1-2\nu}\left(\ln J\right)^2\right].$$

Here, $\mathbf{C}$ is the right Cauchy-Green deformation tensor and J is the determinant of the deformation gradient tensor.

Young's modulus depends on ρ_r^σ according to a power law [23, 24],

$$E_Y = E_0\left(\frac{\rho_r^\sigma}{\rho_0}\right)^\gamma.$$

This type of material references a solid-bound molecule that belongs to a multiphasic mixture. Therefore this material may only be used as the solid (or a component of the solid) in a multiphasic mixture (Section 4.12). The solid-bound molecule must be defined in the <Globals> section (Section 3.4.3) and must be included in the multiphasic mixture using a <solid_bound> tag. The parameter *sbm* must refer to the global index of that solid-bound molecule. The value of ρ_r^σ is specified within the <solid_bound> tag. If a chemical reaction is defined within that multiphasic mixture that alters the value of ρ_r^σ, lower and upper bounds may be specified for this referential density within the <solid_bound> tag to prevent E_Y from reducing to zero or achieving excessively elevated values.

Example:

```
<material id="1" name="solid matrix" type="multiphasic">
  <phi0>0</phi0>
  <solid_bound sbm="1">
    <rho0>1</rho0>
    <rhomin>0.1</rhomin>
    <rhomax>5</rhomax>
  </solid_bound>
  <fixed_charge_density>0</fixed_charge_density>
  <solid type="Carter-Hayes">
    <sbm>1</sbm>
    <E0>10000</E0>
    <rho0>1</rho0>
```

```
      <gamma>2.0</gamma>
      <v>0</v>
    </solid>
    <permeability type="perm-const-iso">
      <perm>1</perm>
    </permeability>
    <osmotic_coefficient type="osm-coef-const">
      <osmcoef>1</osmcoef>
    </osmotic_coefficient>
    <reaction name="solid remodeling" type="mass-action-forward">
      <Vbar>0</Vbar>
      <vP sbm="1">1</vP>
      <forward_rate type="Huiskes reaction rate">
        <B>1</B>
        <psi0>0.01</psi0>
      </forward_rate>
    </reaction>
  </material>
```

4.1.4.3 Cell Growth

The material type for cell growth is "*cell growth*" [10]. The following material parameters need to be defined:

<phir>	intracellular solid volume fraction in reference (strain-free) configuration, φ_r^s	[ ]
<cr>	intracellular molar content of membrane-impermeant solute (moles per volume of the cell in the reference configuration), c_r	[**n**/**L**^{3}]
<ce>	extracellular osmolarity, c_e	[**n**/**L**^{3}]

The Cauchy stress for this material is

$$\boldsymbol{\sigma} = -\pi \mathbf{I},$$

where π is the osmotic pressure, given by

$$\pi = RT\left(\frac{c_r}{J-\varphi_r^s} - c_e\right),$$

where $J = \det \mathbf{F}$ is the relative volume. The values of the universal gas constant R and absolute temperature T must be specified as global constants.

Cell growth may be modeled by simply increasing the mass of the intracellular solid matrix and membrane-impermeant solute. This is achieved by allowing the parameters φ_r^s and c_r to increase over time as a result of growth, by associating them with user-defined load curves. Since cell growth is often accompanied by cell division, and since daughter cells typically achieve the same solid and solute content as their parent, it may be convenient to assume that φ_r^s and c_r increase proportionally, though this is not an obligatory relationship. To ensure that the initial configuration is a stress-free reference configuration, let $c_r = (1-\varphi_r^s)\, c_e$ in the initial state prior to growth.

Example (using units of mm, N, s, nmol, K):

```
<Globals>
  <Constants>
    <T>298</T>
    <R>8.314e-06</R>
    <Fc>0</Fc>
  </Constants>
</Globals>
<Material>
  <material id="1" name="Cell" type="cell growth">
    <phir lc="1">1</phir>
    <cr lc="2">1</cr>
    <ce>300</ce>
  </material>
</Material>
<LoadData>
  <loadcurve id="1" type="smooth">
```

```
      <loadpoint>0,0.3</loadpoint>
      <loadpoint>1,0.6</loadpoint>
    </loadcurve>
    <loadcurve id="2" type="smooth">
      <loadpoint>0,210</loadpoint>
      <loadpoint>1,420</loadpoint>
    </loadcurve>
  </LoadData>
```

4.1.4.4 Kinematic Growth

The material type for kinematic growth is *kinematic growth*. The following parameters must be defined:

<elastic>	Specification of the elastic material undergoing kinematic growth	
<growth>	Specification of the growth model (*volume growth*, *area growth*, *fiber growth*)	

Kinematic growth, based on the framework first proposed in [67], uses the multiplicative decomposition of the deformation gradient into

$$\mathbf{F} = \mathbf{F}^e \cdot \mathbf{F}^g \,,$$

where $\mathbf{F}^e$ represents the elastic deformation and $\mathbf{F}^g$ represents the growth. The related Jacobians are

$$J^e = \det \mathbf{F}^e \quad J^g = \det \mathbf{F}^g \,.$$

A constitutive model for the growth tensor was provided by [59],

$$\mathbf{F}^g = \vartheta^{\text{iso}}\mathbf{I} + \left(\vartheta^{\text{ani}} - 1\right) \mathbf{n}_r \otimes \mathbf{n}_r$$

where $\mathbf{n}_r$ is the referential unit vector for a fiber direction (or normal to an area), ϑ^{iso} represents isotropic growth, and ϑ^{ani} represents anisotropic growth.

For *volume growth*, let $\vartheta^{\text{iso}} = \vartheta^g$ and $\vartheta^{\text{ani}} = 1$, where ϑ^g is called the growth *multiplier* in FEBio, which is typically prescribed using a loadcurve. For *area growth* in the plane transverse to the normal direction $\mathbf{n}$, let $\vartheta^{\text{iso}} = \sqrt{\vartheta^g}$ and $\vartheta^{\text{ani}} = 2 - \sqrt{\vartheta^g}$. Finally, for *fiber growth*, let $\vartheta^{\text{iso}} = 1$ and $\vartheta^{\text{ani}} = \vartheta^g$.

4.1.4.5 Cubic CLE

The material type for a conewise linear elastic (CLE) material with cubic symmetry is *cubic CLE*. The following parameters must be defined:

<lp1>	Tensile diagonal first Lamé coefficient λ_{+1}	[**P**]
<lm1>	Compressive diagonal first Lamé coefficient λ_{-1}	[**P**]
<l2>	Off-diagonal first Lamé coefficient λ_2	[**P**]
<mu>	Second Lamé coefficient μ	[**P**]

This bimodular elastic material is specialized from the orthotropic conewise linear elastic material described by Curnier et al. [28], to the case of cubic symmetry. It is derived from the following hyperelastic strain-energy function:

$$\Psi_r = \mu\mathbf{I} : \mathbf{E}^2 + \sum_{a=1}^{3} \frac{1}{2}\lambda_1 \left[\mathbf{A}_a^r : \mathbf{E}\right] \left(\mathbf{A}_a^r : \mathbf{E}\right)^2 + \frac{1}{2}\lambda_2 \sum_{\substack{b=1 \\ b \neq a}}^{3} \left(\mathbf{A}_a^r : \mathbf{E}\right) \left(\mathbf{A}_b^r : \mathbf{E}\right) \,,$$

where

$$\lambda_1 \left[\mathbf{A}_a^r : \mathbf{E}\right] = \begin{cases} \lambda_{+1} & \mathbf{A}_a^r : \mathbf{E} \geqslant 0 \\ \lambda_{-1} & \mathbf{A}_a^r : \ .E < 0 \end{cases}$$

Here, $\mathbf{E}$ is the Lagrangian strain tensor and $\mathbf{A}_a^r = \mathbf{a}_a^r \otimes \mathbf{a}_a^r$, where $\mathbf{a}_a^r$ ($a = 1, 2, 3$) are orthonormal vectors aligned with the material axes. This material response was originally formulated for infinitesimal strain analyses; its behavior under finite strains may not be physically realistic.

Example:

```
<material id="1" type="cubic CLE">
  <density>1</density>
  <lp1>13.01</lp1>
  <lm1>0.49</lm1>
  <l2>0.66</l2>
  <mu>0.16</mu>
</material>
```

4.1.4.6 Donnan Equilibrium Swelling

The material type for a Donnan equilibrium swelling pressure is "*Donnan equilibrium*". The swelling pressure is described by the equations for ideal Donnan equilibrium, assuming that the material is porous, with a charged solid matrix, and the external bathing environment consists of a salt solution of monovalent counter-ions [62, 51]. Since osmotic swelling must be resisted by a solid matrix, this material is not stable on its own. It must be combined with an elastic material that resists the swelling, using a "*solid mixture*" container as described in Section 4.1.4.26. The following material parameters need to be defined:

<phiw0>	gel porosity (fluid volume fraction) in reference (strain-free) configuration, φ_0^w	[ ]
<cF0>	fixed-charge density in reference (strain-free) configuration, c_0^F	[**n**/**L**^{3}]
<bosm>	external bath osmolarity, $\bar{c}^*$	[**n**/**L**^{3}]
<Phi>	osmotic coefficient, Φ	[]

The Cauchy stress for this material is the stress from the Donnan equilibrium response [6]:

$$\boldsymbol{\sigma} = -\pi \mathbf{I},$$

where π is the osmotic pressure, given by

$$\pi = RT\Phi\left(\sqrt{(c^F)^2 + (\bar{c}^*)^2} - \bar{c}^*\right),$$

and c^F is the fixed-charge density in the current configuration, related to the reference configuration via

$$c^F = \frac{\varphi_0^w}{J - 1 + \varphi_0^w} c_0^F,$$

where $J = \det \mathbf{F}$ is the relative volume. The values of the universal gas constant R and absolute temperature T must be specified as global constants. For ideal Donnan law, use $\Phi = 1$.

Note that c_0^F may be negative or positive; the gel porosity is unitless and must be in the range $0 < \varphi_0^w < 1$. A self-consistent set of units must be used for this model. For example:

	(m, N, s, mol, K)	(mm, N, s, nmol, K)
R	8.314 J/mol·K	8.314×10^{-6} mJ/nmol·K
T	K	K
c_0^F	Eq/m^3= mEq/L	nEq/mm^3= mEq/L
$\bar{c}^*$	mol/m^3= mM	nmol/mm^3= mM
ξ_i	Pa	MPa
π	Pa	MPa

Though this material is porous, this is not a full-fledged biphasic material as described in Section 4.10 for example. The behavior described by this material is strictly valid only after the transient response of interstitial fluid and ion fluxes has subsided (thus Donnan *equilibrium*).

Donnan osmotic swelling reduces to zero when either $c_0^F = 0$ or $\bar{c}^* \to \infty$. Therefore, entering any other values for c_0^F and $\bar{c}^*$ at the initial time point of an analysis produces an instantaneous, non-zero swelling pressure. Depending on the magnitude of this pressure relative to the solid

matrix stiffness, the nonlinear analysis may not converge due to this sudden swelling. Therefore, it is recommended to prescribe a load curve for either <cF0> or <bosm>, to ease into the initial swelling prior to the application of other loading conditions.

Example (using units of mm, N, s, nmol, K):

```
<material id="1" type="solid mixture">
  <mat_axis type="local">0,0,0</mat_axis>
  <solid type="Donnan equilibrium">
    <phiw0>0.8</phiw0>
    <cF0 lc="1">1</cF0>
    <bosm>300</bosm>
  </solid>
  <solid type="ellipsoidal fiber distribution">
    <ksi>0.01,0.01,0.01</ksi>
    <beta>3,3,3</beta>
  </solid>
</material>
<LoadData>
  <loadcurve id="1">
    <loadpoint>0,0</loadpoint>
    <loadpoint>1,150</loadpoint>
  </loadcurve>
</LoadData>
<Globals>
  <Constants>
    <R>8.314e-6</R>
    <T>310</T>
  </Constants>
</Globals>
```

4.1.4.7 Ellipsoidal Fiber Distribution

The material type for an ellipsoidal continuous fiber distribution is "*ellipsoidal fiber distribution*". Since fibers can only sustain tension, this material is not stable on its own. It must be combined with a stable compressible material that acts as a ground matrix, using a "*solid mixture*" container as described in Section 4.1.4.26. The following material parameters need to be defined:

<beta>	parameters $(\beta_1, \beta_2, \beta_3)$	[ ]
<ksi>	parameters (ξ_1, ξ_2, ξ_3)	[**P**]

The Cauchy stress for this fibrous material is given by [52, 4, 6]:

$$\boldsymbol{\sigma} = \int_0^{2\pi} \int_0^{\pi} H\left(I_n - 1\right) \sigma_n\left(\mathbf{n}\right) \sin\varphi \, d\varphi \, d\theta.$$

Here, $I_n = \lambda_n^2 = \mathbf{N} \cdot \mathbf{C} \cdot \mathbf{N}$ is the square of the fiber stretch λ_n, $\mathbf{N}$ is the unit vector along the fiber direction, in the reference configuration, which in spherical angles is directed along (θ, φ), $\mathbf{n} = \mathbf{F} \cdot \mathbf{N}/\lambda_n$, and $H\left(.\right)$is the unit step function that enforces the tension-only contribution.

The fiber stress is determined from a fiber strain energy function,

$$\sigma_n = \frac{2I_n}{J} \frac{\partial \Psi}{\partial I_n} \mathbf{n} \otimes \mathbf{n},$$

where in this material,

$$\Psi\left(\mathbf{n}, I_n\right) = \xi\left(\mathbf{n}\right)\left(I_n - 1\right)^{\beta(\mathbf{n})}.$$

The materials parameters $\beta$and $\xi$are assumed to vary ellipsoidally with $\mathbf{n}$, according to

$$\xi\left(\mathbf{n}\right) = \left(\frac{\cos^2\theta \sin^2\varphi}{\xi_1^2} + \frac{\sin^2\theta \sin^2\varphi}{\xi_2^2} + \frac{\cos^2\varphi}{\xi_3^2}\right)^{-1/2}$$
$$\beta\left(\mathbf{n}\right) = \left(\frac{\cos^2\theta \sin^2\varphi}{\beta_1^2} + \frac{\sin^2\theta \sin^2\varphi}{\beta_2^2} + \frac{\cos^2\varphi}{\beta_3^2}\right)^{-1/2}.$$

The orientation of the material axis can be defined as explained in detail in Section 4.1.1.

Example:

```
<material id="1" type="solid mixture">
  <mat_axis type="local">0,0,0</mat_axis>
  <solid type="neo-Hookean">
    <E>1000.0</E>
    <v>0.45</v>
  </solid>
  <solid type="ellipsoidal fiber distribution">
    <ksi>10, 12, 15</ksi>
    <beta>2.5, 3, 3</beta>
  </solid>
</material>
```

4.1.4.8 Ellipsoidal Fiber Distribution Neo-Hookean

The material type for a Neo-Hookean material with an ellipsoidal continuous fiber distribution is "*EFD neo-Hookean*". The following material parameters need to be defined:

<E>	Young's modulus	[**P**]
<v>	Poisson's ratio	[ ]
<beta>	parameters $(\beta_1, \beta_2, \beta_3)$	[ ]
<ksi>	parameters (ξ_1, ξ_2, ξ_3)	[**P**]

The Cauchy stress for this material is given by,

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}_{NH} + \boldsymbol{\sigma}_f.$$

Here, $\boldsymbol{\sigma}_{NH}$ is the stress from the Neo-Hookean basis (Section 4.1.4.19), and $\boldsymbol{\sigma}_f$ is the stress contribution from the fibers (Section 4.1.4.7).

Example:

```
<material id="1" type="EFD neo-Hookean">
  <E>1</E>
  <v>0.3</v>
  <beta>4.5,4.5,4.5</beta>
  <ksi>1,1,1</ksi>
  <mat_axis type="local">0,0,0</mat_axis>
</material>
```

4.1.4.9 Ellipsoidal Fiber Distribution with Donnan Equilibrium Swelling

The material type for a swelling pressure combined with an ellipsoidal continuous fiber distribution is "*EFD Donnan equilibrium*". The swelling pressure is described by the equations for ideal Donnan equilibrium, assuming that the material is porous, with a charged solid matrix, and the external bathing environment consists of a salt solution of monovalent counter-ions. The following material parameters need to be defined:

<phiw0>	gel porosity (fluid volume fraction) in reference (strain-free) configuration, φ_0^w	[ ]
<cF0>	fixed-charge density in reference (strain-free) configuration, c_0^F	[**n**/**L**^{3}]
<bosm>	external bath osmolarity, $\overline{c}^*$	[**n**/**L**^{3}]
<beta>	parameters $(\beta_1, \beta_2, \beta_3)$	[ ]
<ksi>	parameters (ξ_1, ξ_2, ξ_3)	[**P**]

The Cauchy stress for this material is given by,

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}_{DE} + \boldsymbol{\sigma}_f \,.$$

$\boldsymbol{\sigma}_f$ is the stress contribution from the fibers, as described in Section 4.1.1. $\boldsymbol{\sigma}_{DE}$ is the stress from the Donnan equilibrium response, as described in Section 4.1.4.6

Example (using units of mm, N, s, nmol, K):

```
<material id="1" type="EFD Donnan equilibrium">
  <phiw0>0.8</phiw0>
  <cF0 lc="1">1</cF0>
  <bosm>300</bosm>
  <beta>3,3,3</beta>
  <ksi>0.01,0.01,0.01</ksi>
  <mat_axis type="local">0,0,0</mat_axis>
</material>
<LoadData>
  <loadcurve id="1">
    <loadpoint>0,0</loadpoint>
    <loadpoint>1,150</loadpoint>
  </loadcurve>
</LoadData>
<Globals>
  <Constants>
    <R>8.314e-6</R>
    <T>310</T>
  </Constants>
</Globals>
```

4.1.4.10 Fung Orthotropic Compressible

The material type for unconstrained orthotropic Fung elasticity [33, 32] is "*Fung-ortho-compressible*". The following material parameters must be defined:

<E1>	E_1 Young's modulus	[**P**]
<E2>	E_2 Young's modulus	[**P**]
<E3>	E_3 Young's modulus	[**P**]
<G12>	G_{12} shear modulus	[**P**]
<G23>	G_{23} shear modulus	[**P**]
<G31>	G_{31} shear modulus	[**P**]
<v12>	ν_{12} Poisson's ratio	[ ]
<v23>	ν_{23} Poisson's ratio	[ ]
<v31>	ν_{31} Poisson's ratio	[ ]
<c>	c coefficient	[**P**]
<k>	κ bulk modulus	[**P**]

The hyperelastic strain energy function is given by [5],

$$\Psi = \frac{1}{2}c\left(e^{Q}-1\right)+U\left(J\right),$$

where,

$$Q = c^{-1}\sum_{a=1}^{3}\left[2\mu_a\mathbf{M}_a:\mathbf{E}^2+\sum_{b=1}^{3}\lambda_{ab}\left(\mathbf{M}_a:\mathbf{E}\right)\left(\mathbf{M}_b:\mathbf{E}\right)\right],$$

and

$$U\left(J\right) = \frac{\kappa}{2}\left(\ln J\right)^2 .$$

Here, $\mathbf{E} = \left(\mathbf{C}-\mathbf{I}\right)/2$ and $\mathbf{M}_a = \mathbf{V}_a \otimes \mathbf{V}_a$where $\mathbf{V}_a$ defines the initial direction of material axis a. See Section 4.1.1.2 on how to define the material axes for orthotropic materials. The Lamé constants μ_a $(a = 1, 2, 3)$ and λ_{ab} $(a, b = 1, 2, 3,\ \lambda_{ba} = \lambda_{ab})$ are related to Young's moduli E_a, shear moduli G_{ab} and Poisson's ratios ν_{ab} via

$$\begin{bmatrix} \lambda_{11}+2\mu_1 & \lambda_{12} & \lambda_{13} & 0 & 0 & 0 \\ \lambda_{12} & \lambda_{22}+2\mu_2 & \lambda_{23} & 0 & 0 & 0 \\ \lambda_{13} & \lambda_{23} & \lambda_{33}+2\mu_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2}\left(\mu_1+\mu_2\right) & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2}\left(\mu_2+\mu_3\right) & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2}\left(\mu_1+\mu_3\right) \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} \frac{1}{E_1} & -\frac{\nu_{12}}{E_1} & -\frac{\nu_{13}}{E_1} & 0 & 0 & 0 \\ -\frac{\nu_{21}}{E_2} & \frac{1}{E_2} & -\frac{\nu_{23}}{E_2} & 0 & 0 & 0 \\ -\frac{\nu_{31}}{E_3} & -\frac{\nu_{32}}{E_3} & \frac{1}{E_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{12}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{23}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{31}} \end{bmatrix}$$

The orthotropic Lamé parameters should produce a positive definite stiffness matrix.

Example:

```
<material id="3" type="Fung-ortho-compressible">
  <E1>124</E1>
  <E2>124</E2>
  <E3>36</E3>
  <G12>67</G12>
  <G23>40</G23>
  <G31>40</G31>
  <v12>-0.075</v12>
  <v23>0.87</v23>
  <v31>0.26</v31>
  <c>1</c>
  <k>120</k>
</material>
```

4.1.4.11 Gent Compressible

The compressible Gent material is defined using the *"compressible Gent"* type string. It defines the following parameters.

<G>	Shear modulus	[**P**]
<Jm>	$J_m = I_m - 1$, with I_m max value for first invariant I_1	[**P**]
<k>	Bulk modulus	[**P**]

The strain energy of this material is defined as follows,

$$\Psi_r = -\frac{\mu J_m}{2} \ln\left(1 - \frac{I_1 - 3}{J_m}\right) + \frac{\kappa}{2}\left(\frac{J^2 - 1}{2} - \ln J\right)^4$$

Here, μ is the shear modulus.
Example:

```
<material id="2" type="compressible Gent">
  <G>3.14</G>
  <Jm>1.5</Jm>
  <k>1e5</k>
</material>
```

4.1.4.12 Holmes-Mow

The material type for the Holmes-Mow material [42] is *Holmes-Mow*. This isotropic hyperelastic material has been used to represent the solid matrix of articular cartilage [42, 8] and intervertebral disc [46]. The following material parameters must be defined:

<E>	Young's modulus	[**P**]
<v>	Poisson's ratio	[ ]
<beta>	Exponential stiffening coefficient	[ ]

The coupled hyperelastic strain-energy function for this material is given by [42]:

$$W\left(I_1, I_2, J\right) = \frac{1}{2} c \left(e^Q - 1\right),$$

where I_1 and I_2 are the first and second invariants of the right Cauchy-Green tensor and $J$is the jacobian of the deformation gradient. Furthermore,

$$\begin{aligned} Q &= \frac{\beta}{\lambda + 2\mu}\left[\left(2\mu - \lambda\right)\left(I_1 - 3\right) + \lambda\left(I_2 - 3\right) - \left(\lambda + 2\mu\right)\ln J^2\right] \\ c &= \frac{\lambda + 2\mu}{2\beta} \end{aligned},$$

and λ and μ are the Lamé parameters which are related to the more familiar Young's modulus and Poisson's ratio in the usual manner:

$$\begin{aligned} \lambda &= \frac{E}{\left(1 + \nu\right)\left(1 - 2\nu\right)} \\ \mu &= \frac{E}{2\left(1 + \nu\right)} \end{aligned}.$$

Example:

```
<material id="3" type="Holmes-Mow">
  <E>1</E>
  <v>0.35</v>
  <beta>0.25</beta>
</material>
```

4.1.4.13 Holzapfel-Gasser-Ogden Unconstrained

The material type for the unconstrained Holzapfel-Gasser-Ogden material [35] is *HGO unconstrained*. The following material parameters must be defined:

<c>	Shear modulus of ground matrix	[**P**]
<k1>	Fiber modulus	[**P**]
<k2>	Fiber exponential coefficient	[**P**]
<gamma>	Fiber mean orientation angle γ	[**deg**]
<kappa>	Fiber dispersion	[]
<k>	Bulk modulus	[**P**]

The strain-energy function is given by:

$$\Psi_r = \frac{c}{2}\left(I_1 - 3\right) - c\ln J + \frac{k_1}{2k_2}\sum_{\alpha}\left(\exp\left(k_2\left\langle E_\alpha\right\rangle^2\right) - 1\right) + \frac{K_0}{2}\left(\frac{J^2-1}{2} - \ln J\right)$$

The fiber strain is

$$E_\alpha = \kappa\left(I_1 - 3\right) + \left(1 - 3\kappa\right)\left(I_{4\alpha} - 1\right)$$

where $I_1 = \operatorname{tr}\mathbf{C}$ and $I_{4\alpha} = \mathbf{a}_{\alpha r}\cdot\mathbf{C}\cdot\mathbf{a}_{\alpha r}$. The Macaulay brackets around $\left\langle\tilde{E}_\alpha\right\rangle$ indicate that this term is zero when $E_\alpha < 0$ and equal to E_α when this strain is positive.

There are two fiber families along the vectors $\mathbf{a}_{\alpha r}$ ($\alpha = 1, 2$), lying in the $\{\mathbf{e}_1, \mathbf{e}_2\}$ plane of the local material axes $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$, making an angle $\pm\gamma$ with respect to $\mathbf{e}_1$. Each fiber family has a dispersion κ, where $0 \le \kappa \le \frac{1}{3}$. When $\kappa = 0$ there is no fiber dispersion, implying that all the fibers in that family act along the angle $\pm\gamma$; the value $\kappa = \frac{1}{3}$ represents an isotropic fiber dispersion. All other intermediate values of κ produce a $\pi-$periodic von Mises fiber distribution, as described in [35]. c is the shear modulus of the ground matrix; k_1 is the fiber modulus and k_2 is the exponential coefficient.

Unlike the uncoupled Holzapfel-Gasser-Ogden material presented in Section 4.1.2.8, this unconstrained version does not enforce isochoric deformation. This unconstrained model may be used to describe the porous solid matrix of a biphasic or multiphasic tissue model, where pore volume may change in response to influx or efflux of interstitial fluid.

Example:

```
<material id="2" type="HGO unconstrained">
  <c>7.64</c>
  <k1>996.6</k1>
  <k2>524.6</k2>
  <gamma>49.98</gamma>
  <kappa>0.226</kappa>
  <k>7.64e3</k>
</material>
```

4.1.4.14 Isotropic Elastic

The material type for isotropic elasticity is *isotropic elastic*[1]. The following material parameters must be defined:

<E>	Young's modulus	[**P**]
<v>	Poisson's ratio	[ ]

This material is an implementation of a hyperelastic constitutive material that reduces to the classical linear elastic material for small strains, but is objective for large deformations and rotations. The hyperelastic strain-energy function is given by:

$$W = \frac{1}{2}\lambda \left(\operatorname{tr} \mathbf{E}\right)^2 + \mu \mathbf{E} : \mathbf{E}.$$

Here, $\mathbf{E}$ is the Euler-Lagrange strain tensor and λ and μ are the Lamé parameters, which are related to the more familiar Young's modulus E and Poisson's ratio ν as follows:

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \mu = \frac{E}{2(1+\nu)}.$$

It is often convenient to express the material properties using the bulk modulus K and shear modulus G. To convert to Young's modulus and Poisson's ratio, use the following formulas:

$$E = \frac{9KG}{3K+G}, \quad \nu = \frac{3K-2G}{6K+2G}.$$

Remark: Note that although this material is objective, it is not advised to use this model for large strains since the behavior may be unphysical. For example, it can be shown that for a uniaxial tension the stress grows unbounded and the volume tends to zero for finite strains. Also for large strains, the Young's modulus and Poisson's ratio input values have little relationship to the actual material parameters. Therefore it is advisable to use this material only for small strains and/or large rotations. To represent isotropic elastic materials under large strain and rotation, it is best to use some of the other available nonlinear material models described in this chapter, such as the Holmes-Mow material in Section 4.1.4.12, the neo-Hookean material in Section 4.1.4.19, or the unconstrained Ogden material in Section 4.1.4.22.

Example:

```
<material id="1" type="isotropic elastic">
  <E>1000.0</E>
  <v>0.45</v>
</material>
```

[1] This material replaces the now-obsolete *linear elastic* and *St.Venant-Kirchhoff* materials. These materials are still available for backward compatibility although it is recommended to use the *isotropic elastic* material instead.

4.1.4.15 Orthotropic Elastic

The material type for orthotropic elasticity is "*orthotropic elastic*". The following material parameters must be defined:

<E1>	Young's modulus in the x-direction	[**P**]
<E2>	Young's modulus in the y-direction	[**P**]
<E3>	Young's modulus in the z-direction	[**P**]
<G12>	Shear modulus in the xy-plane	[**P**]
<G23>	Shear modulus in the yz-plane	[**P**]
<G31>	Shear modulus in the xz-plane	[**P**]
<v12>	Poisson's ratio between x- and y-direction	[ ]
<v23>	Poisson's ratio between y- and z-direction	[ ]
<v31>	Poisson's ratio between z- and x-direction	[ ]

The stress-strain relation for this material is given by

$$\begin{bmatrix} E_{11} \\ E_{22} \\ E_{33} \\ 2E_{23} \\ 2E_{31} \\ 2E_{12} \end{bmatrix} = \begin{bmatrix} 1/E_1 & -\nu_{21}/E_2 & -\nu_{31}/E_3 & 0 & 0 & 0 \\ -\nu_{12}/E_1 & 1/E_2 & -\nu_{32}/E_3 & 0 & 0 & 0 \\ -\nu_{13}/E_1 & -\nu_{23}/E_2 & 1/E_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/G_{23} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/G_{31} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/G_{12} \end{bmatrix} \begin{bmatrix} T_{11} \\ T_{22} \\ T_{33} \\ T_{23} \\ T_{31} \\ T_{12} \end{bmatrix}$$

Material axes may be specified as described in Section 4.1.1.2.

Example:

```
<material id="3" type="orthotropic elastic">
  <mat_axis type="vector">
    <a>0.866,0.5,0</a>
    <d>-0.5,0.866,0</d>
  </mat_axis>
  <E1>1</E1>
  <E2>2</E2>
  <E3>3</E3>
  <v12>0</v12>
  <v23>0</v23>
  <v31>0</v31>
  <G12>1</G12>
  <G23>1</G23>
  <G31>1</G31>
</material>
```

4.1.4.16 Orthotropic CLE

The material type for a conewise linear elastic (CLE) material with orthtropic symmetry is *orthotropic CLE*. The following parameters must be defined:

<lp11>	Tensile diagonal first Lamé coefficient along direction 1 λ_{+11}	[P]
<lp22>	Tensile diagonal first Lamé coefficient along direction 2 λ_{+22}	[P]
<lp11>	Tensile diagonal first Lamé coefficient along direction 3 λ_{+33}	[P]
<lm11>	Compressive diagonal first Lamé coefficient along direction 1 λ_{-11}	[P]
<lm22>	Compressive diagonal first Lamé coefficient along direction 2 λ_{-22}	[P]
<lm33>	Compressive diagonal first Lamé coefficient along direction 3 λ_{-33}	[P]
<l12>	Off-diagonal first Lamé coefficient in 1-2 plane λ_{12}	[P]
<l23>	Off-diagonal first Lamé coefficient in 2-3 plane λ_{23}	[P]
<l31>	Off-diagonal first Lamé coefficient in 3-1 plane λ_{31}	[P]
<mu1>	Second Lamé coefficient along direction 1 μ_1	[P]
<mu2>	Second Lamé coefficient along direction 2 μ_2	[P]
<mu3>	Second Lamé coefficient along direction 3 μ_3	[P]

This bimodular elastic material is the orthotropic conewise linear elastic material described by Curnier et al. [28]. It is derived from the following hyperelastic strain-energy function:

$$\Psi_r = \sum_{a=1}^{3} \mu_a \mathbf{A}_a^r : \mathbf{E}^2 + \frac{1}{2}\lambda_{aa}\left[\mathbf{A}_a^r : \mathbf{E}\right]\left(\mathbf{A}_a^r : \mathbf{E}\right)^2 + \sum_{\substack{b=1 \\ b \neq a}}^{3} \frac{1}{2}\lambda_{ab}\left(\mathbf{A}_a^r : \mathbf{E}\right)\left(\mathbf{A}_b^r : \mathbf{E}\right)$$

where $\lambda_{ba} = \lambda_{ab}$ and

$$\lambda_{aa}\left[\mathbf{A}_a^r : \mathbf{E}\right] = \begin{cases} \lambda_{+aa} & \mathbf{A}_a^r : \mathbf{E} \geqslant 0 \\ \lambda_{-aa} & \mathbf{A}_a^r : \mathbf{E} < 0 \end{cases}, \quad a = 1, 2, 3$$

Here, $\mathbf{E}$ is the Lagrangian strain tensor and $\mathbf{A}_a^r = \mathbf{a}_a^r \otimes \mathbf{a}_a^r$, where $\mathbf{a}_a^r$ $(a = 1, 2, 3)$ are orthonormal vectors aligned with the material axes. This material response was originally formulated for infinitesimal strain analyses; its behavior under finite strains may not be physically realistic.

Example:

```
<material id="1" type=" orthotropic CLE">
  <density>1</density>
  <lp11>13.01</lp11>
  <lp22>13.01</lp22>
  <lp33>13.01</lp33>
  <lm11>0.49</lm11>
  <lm22>0.49</lm22>
  <lm33>0.49</lm33>
  <l12>0.66</l12>
  <l23>0.66</l23>
  <l31>0.66</l31>
  <mu1>0.16</mu1>
  <mu2>0.16</mu2>
  <mu3>0.16</mu3>
</material>
```

4.1.4.17 Osmotic Pressure from Virial Expansion

The material type for osmotic pressure from virial expansion is "*osmotic virial expansion*". The following material parameters need to be defined:

<phiw0>	Fluid volume fraction in reference (strain-free) configuration, φ_r^w	[ ]
<cr>	Concentration of interstitial solute causing the osmotic pressure (moles per volume of the mixture in the reference configuration), c_r	[**n**/**L**^{3}]
<c1>	First virial coefficient c_1	[**F**·**L**/**n**]
<c2>	Second virial coefficient c_2	[**F**·**L**^{4}/**n**^{2}]
<c3>	Third virial coefficient c_3	[**F**·**L**^{7}/**n**^{3}]

The Cauchy stress for this material is

$$\boldsymbol{\sigma} = -\pi \mathbf{I},$$

where π is the osmotic pressure, given by

$$\pi = c_1 c + c_2 c^2 + c_3 c^3, \quad c = \frac{\varphi_r^w c_r}{J - 1 + \varphi_r^w},$$

c is the solute concentration in the current configuration, and $J = \det \mathbf{F}$ is the relative volume.

This osmotic swelling pressure in the interstitial fluid of a porous material represents an entropic mechanism whose magnitude is independent of the external bath osmolarity. Typically, this material should be used in a solid mixture where the swelling pressure is resisted by a solid matrix in tension.

Example:

```
<Material>
  <material id="1" type="solid mixture">
    <solid type="osmotic virial expansion">
      <phiw0>0.8</phiw0>
      <cr lc="1">100</cr>
      <c1>2.436e-6</c1>
      <c2>0</c2>
      <c3>0</c3>
    </solid>
    <solid type="spherical fiber distribution"/>
  </material>
</Material>
<LoadData>
  <loadcurve id="1" type="smooth">
    <loadpoint>0,0</loadpoint>
    <loadpoint>1,1</loadpoint>
  </loadcurve>
</LoadData>
```

4.1.4.18 Natural Neo-Hookean

The material type for a natural Neo-Hookean material is *natural neo-Hookean*. The following parameters must be defined:

<E>	Young's modulus E	[**P**]
<v>	Poisson's ratio ν	[ ]

This model describes an unconstrained Neo-Hookean material using the natural strain tensor [27]. It has a non-linear stress-strain behavior, but reduces to the classical linear elasticity model for small strains *and* small rotations. It is derived from the following hyperelastic strain-energy function:

$$W = \frac{\kappa}{2} K_1^2 + \mu K_2^2,$$

where κ is the bulk modulus and μ is the shear modulus,

$$\kappa = \frac{E}{3\,(1-2\nu)}\,, \quad \mu = \frac{E}{2\,(1+\nu)}$$

Here, K_1 and K_2 are the first and second invariants of the left natural (Hencky) strain tensor $\boldsymbol{\eta} = \ln \mathbf{V}$ where $\mathbf{V}$ is the left stretch tensor in the polar decomposition $\mathbf{F} = \mathbf{V} \cdot \mathbf{R}$.

This material model uses a standard displacement-based element formulation, so care must be taken when modeling materials with nearly-incompressible material behavior to avoid element locking. For this case, use the *Mooney-Rivlin* material described in Section 4.1.2.9.

Example:

```
<material id="1" type="natural neo-Hookean">
  <E>200e3</E>
  <v>0.3</v>
</material>
```

4.1.4.19 Neo-Hookean

The material type for a Neo-Hookean material is *neo-Hookean*. The following parameters must be defined:

<E>	Young's modulus	[**P**]
<v>	Poisson's ratio	[ ]

This model describes an unconstrained Neo-Hookean material [22]. It has a non-linear stress-strain behavior, but reduces to the classical linear elasticity model for small strains *and* small rotations. It is derived from the following hyperelastic strain-energy function:

$$W = \frac{\mu}{2}\left(I_1 - 3\right) - \mu \ln J + \frac{\lambda}{2}\left(\ln J\right)^2 .$$

Here, I_1 and I_2 are the first and second invariants of the right Cauchy-Green deformation tensor $\mathbf{C}$ and J is the determinant of the deformation gradient tensor. The relationship between the material parameters, E, and v, and the parameters used in the strain-energy function, is as follows.

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

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

This material model uses a standard displacement-based element formulation, so care must be taken when modeling materials with nearly-incompressible material behavior to avoid element locking. For this case, use the *Mooney-Rivlin* material described in Section 4.1.2.9.

Example:

```
<material id="1" type="neo-Hookean">
  <E>1000.0</E>
  <v>0.45</v>
</material>
```

4.1.4.20 Coupled Mooney-Rivlin

The coupled Mooney-Rivlin material describes an unconstrained formulation of the Mooney-Rivlin material. The material type for this material is *coupled Mooney-Rivlin*. The following material parameters can be defined.

<c1>	Mooney-Rivlin c1 parameter	[**P**]
<c2>	Mooney-Rivlin c2 parameter	[**P**]
k	"Bulk-modulus"	[**P**]

The strain-energy function is given by the following expression.

$$W = c_1 (I_1 - 3) + c_2 (I_2 - 3) - 2 (c_1 + 2c_2) \ln J + \frac{\lambda}{2} (\ln J)^2$$

Here, I_1 and I_2 are the first and second invariants of the right Cauchy-Green deformation tensor $\mathbf{C}$ and J is the determinant of the deformation gradient tensor.

This material model uses a standard displacement-based element formulation, so care must be taken when modeling materials with nearly-incompressible material behavior to avoid element locking. For (nearly-) incompressible materials, use the *Mooney-Rivlin* material described in Section 4.1.2.9.

Example:

```
<material id="1" type="coupled Mooney-Rivlin">
  <c1>10.0</c1>
  <c2>1.0</c2>
  <k>100.0</k>
</material>
```

4.1.4.21 Coupled Veronda-Westmann

The material type for the coupled Veronda-Westmann material is *coupled Veronda-Westmann*. The following material parameters can be defined.

<c1>	Veronda-Westmann c1 parameter	[**P**]
<c2>	Veronda-Westmann c2 parameter	[**P**]
k	"Bulk-modulus"	[**P**]

The coupled Veronda-Westmann material is an unconstrained formulation of the Veronda-Westmann material and is defined by the following strain-energy function.

$$W = c_1 \left(e^{c_2 (I_1 - 3)} - 1 \right) - \frac{c_1 c_2}{2} (I_2 - 3) + \frac{\lambda}{2} (\ln J)^2$$

Here, I_1 and I_2 are the first and second invariants of the right Cauchy-Green deformation tensor **C** and $J$is the determinant of the deformation gradient tensor.

This material model uses a standard displacement-based element formulation, so care must be taken when modeling materials with nearly-incompressible material behavior to avoid element locking. For (nearly-) incompressible materials, use the *Veronda-Westmann* material described in Section 4.1.2.17.

Example:

```
<material id="1" type="coupled Veronda-Westmann">
  <c1>10.0</c1>
  <c2>1.0</c2>
  <k>100.0</k>
</material>
```

4.1.4.22 Ogden Unconstrained

This material describes an unconstrained hyperelastic Ogden material [73]. The following material parameters must be defined:

<c[n]>	Coefficient of n^{th} term, where n can range from 1 to 6	[**P**]
<m[n]>	Exponent of n^{th} term, where n can range from 1 to 6	[ ]
<cp>	Bulk-like modulus	[**P**]

The hyperelastic strain energy function for this material is given in terms of the eigenvalues of the right or left stretch tensor:

$$W(\lambda_1, \lambda_2, \lambda_3) = \frac{1}{2} c_p (J-1)^2 + \sum_{i=1}^{N} \frac{c_i}{m_i^2} \left(\lambda_1^{m_i} + \lambda_2^{m_i} + \lambda_3^{m_i} - 3 - m_i \ln J\right).$$

Here, λ_i^2 are the eigenvalues of the right or left Cauchy deformation tensor, c_p, c_i and $m_i$are material coefficients and N ranges from 1 to 6. Any material parameters that are not specified by the user are assumed to be zero.

Example:

```
<material id="1" type="Ogden unconstrained">
  <m1>2.4</m1>
  <c1>1</c1>
  <cp>2</cp>
</material>
```

4.1.4.23 Perfect Osmometer Equilibrium Osmotic Pressure

The material type for a perfect osmometer equilibrium swelling pressure is "*perfect osmometer*". The swelling pressure is described by the equations for a perfect osmometer, assuming that the material is porous, containing an interstitial solution whose solutes cannot be exchanged with the external bathing environment; similarly, solutes in the external bathing solution cannot be exchanged with the interstitial fluid of the porous material. Therefore, osmotic pressurization occurs when there is an imbalance between the interstitial and bathing solution osmolarities. Since osmotic swelling must be resisted by a solid matrix, this material is not stable on its own. It must be combined with an elastic material that resists the swelling, using a "*solid mixture*" container as described in Section 4.1.4.26. The following material parameters need to be defined:

<phiw0>	gel porosity (fluid volume fraction) in reference (strain-free) configuration, φ_0^w	[ ]
<iosm>	interstitial fluid osmolarity in reference configuration, $\bar{c}_0$	[**n**/**L**^{3}]
<bosm>	external bath osmolarity, $\bar{c}^*$	[**n**/**L**^{3}]

The Cauchy stress for this material is the stress from the perfect osmometer equilibrium response:

$$\boldsymbol{\sigma} = -\pi \mathbf{I},$$

where π is the osmotic pressure, given by

$$\pi = RT\left(\bar{c} - \bar{c}^*\right) .$$

$\bar{c}$ is the interstitial fluid in the current configuration, related to the reference configuration via

$$\bar{c} = \frac{\varphi_0^w}{J - 1 + \varphi_0^w}\bar{c}_0 ,$$

where $J = \det \mathbf{F}$ is the relative volume. The values of the universal gas constant R and absolute temperature T must be specified as global constants.

Though this material is porous, this is not a full-fledged biphasic material as described in Section 4.10 for example. The behavior described by this material is strictly valid only after the transient response of interstitial fluid and solute fluxes has subsided.

Example (using units of mm, N, s, nmol, K):

Hyperosmotic loading of a cell with a membrane-impermeant solute, starting from isotonic conditions.

```
<material id="1" type="solid mixture">
  <solid type="perfect osmometer">
    <phiw0>0.8</ phiw0>
    <iosm>300</cF0>
    <bosm lc="1">1</bosm>
  </solid>
  <solid type="neo-Hookean">
    <E>1.0</E>
    <v>0</v>
  </solid>
```

```
</material>
<LoadData>
  <loadcurve id="1">
    <loadpoint>0,300</loadpoint>
    <loadpoint>1,1500</loadpoint>
  </loadcurve>
</LoadData>
<Globals>
  <Constants>
    <R>8.314e-6</R>
    <T>310</T>
  </Constants>
</Globals>
```

4.1.4.24 Porous Neo-Hookean

The material type for a porous neo-Hookean solid is "*porous neo-Hookean*". This material describes a porous neo-Hookean material with referential solid volume fraction φ^s_r (i.e., porosity $\varphi^w_r = 1 - \varphi^s_r$). The pores are compressible but the skeleton is intrinsically incompressible. Thus, upon pore closure, the material behavior needs to switch from compressible to incompressible. This material may be used to model the porous solid matrix of a biphasic or multiphasic material, in which case it inherits the value of φ^s_r from that parent material.

The material type for a porous Neo-Hookean material is *porous neo-Hookean*. The following parameters must be defined:

<E>	Young's modulus	[**P**]
<phi0>	Referential solid volume fraction ($0 \le \varphi^s_r < 1$)	[ ]

This model is derived from the following hyperelastic strain-energy function:

$$W = \frac{\mu}{2}\left(\bar{I}_1 - 3\right) - \mu \ln \bar{J}\,,$$

where $\bar{J}$ represents the pore volume ratio, evaluated from

$$\bar{J} = \frac{J - \varphi^s_r}{1 - \varphi^s_r}\,,$$

and $\bar{I}_1 = \mathrm{tr}\bar{\mathbf{C}}$, with

$$\bar{\mathbf{F}} = \left(\frac{\bar{J}}{J}\right)^{1/3}\mathbf{F}\,, \quad \bar{\mathbf{C}} = \bar{\mathbf{F}}^T\cdot\bar{\mathbf{F}} = \left(\frac{\bar{J}}{J}\right)^{2/3}\mathbf{C}\,,$$

and $\bar{J} = \det\bar{\mathbf{F}}$. The porosity φ^w in the current configuration evolves with the volume ratio J according to

$$\varphi^w = \frac{J - 1 + \varphi^w_r}{J} = \frac{J - \varphi^s_r}{J} = \bar{J}\left(1 - \varphi^s_r\right)\,.$$

Thus, pore closure occurs when $J = \varphi^s_r$ and $\bar{J} = 0$.

By comparison to a standard neo-Hookean material, this porous neo-Hookean material has an effective Young's modulus equal to

$$E = \frac{3\mu}{1 + \frac{1}{2}\left(\varphi^w_r\right)^2}\,,$$

and an effective Poisson's ratio equal to

$$\nu = \frac{1 - \left(\varphi^w_r\right)^2}{2 + \left(\varphi^w_r\right)^2}\,.$$

Therefore, the two material properties that need to be provided are E and the referential porosity $\varphi^s_r = 1 - \varphi^w_r$. Poisson's ratio in the limit of infinitesimal strains is dictated by the porosity according to the above formula. In particular, a highly porous material ($\varphi^w_r \to 1$) has an effective Poisson ratio that approaches zero ($\nu \to 0$) and $E \to 2\mu$ in the range of infinitesimal strains. A low porosity material ($\varphi^w_r \to 0$) has $\nu \to \frac{1}{2}$ and $E \to 3\mu$, which is the expected behavior of an incompressible neo-Hookean solid. Note that setting $\varphi^w_r = 0$ ($\varphi^s_r = 1$) would not produce good numerical behavior, since the Cauchy stress in an incompressible material would need to be supplemented by a hydrostatic pressure term (a Lagrange multiplier that enforces the incompressibility constraint). Nevertheless, this compressible porous neo-Hookean material behaves well even for values of φ^w as low as ~ 0.015.

Example:

```
<material id="1" type="porous neo-Hookean">
  <density>1.0</density>
  <E>1.0</E>
  <phi0>0.2</phi0>
</material>
```

4.1.4.25 Lung Material

This is a material used for modeling human lung tissue as described by Birzle[18]. The following parameters must be defined:

<E>	Young's modulus, E	[**P**]
<v>	Poisson's ratio, ν	[ ]
<c1>	Birzle-Wall c1 parameter, c_1	[**P**]
<c3>	Birzle-Wall c3 parameter, c_3	[**P**]
<d1>	Birzle-Wall exponential d1 parameter, d_1	[]
<d3>	Birzle-Wall exponential d3 parameter, d_3	[]

This model is derived from the following hyperelastic strain-energy function:

$$W = c\left(I_1 - 3\right) + \frac{c}{\beta}\left(I_3^{-\beta} - 1\right) + c_1\left(I_3^{-\frac{1}{3}} I_1 - 3\right)^{d_1} + c_3\left(I_3^{\frac{1}{3}} - 1\right)^{d_3},$$

where I_1 and I_2 are the first and second invariants of the right Cauchy-Green deformation tensor $\mathbf{C}$,

$$c = \frac{E}{4\left(1+\nu\right)},$$

and

$$\beta = \frac{\nu}{1-2\nu}.$$

Example:

```
<material id="1" type="lung">
  <density>1</density>
  <E>1913.7</E>
  <v>0.3413</v>
  <c1>278.2</c1>
  <c3>5.766</c3>
  <d1>3</d1>
  <d3>6</d3>
</material>
```

4.1.4.26 Solid Mixture

The material type for a constrained mixture of solids is "*solid mixture*". This material describes a mixture of elastic solids that share the same reference configuration and same deformation gradient. It is a container for any combination of the unconstrained elastic materials described in Section 4.1.4.

<solid>	Container tag for unconstrained solid material

The mixture may consist of any number of solids. The stress tensor for the solid mixture is the sum of the stresses for all the solids.

Material axes may be optionally specified within the <material> level, as well as within each <solid>. Within the <material> level, these represent the local element axes relative to the global coordinate system. Within the <solid>, they represent local material axes relative to the element. If material axes are specified at both levels, they are properly compounded to produce local material axes relative to the global coordinate system. Material axes specified in the <ElementData> section are equivalent to a specification at the <material> level: they correspond to local element axes relative to the global system.

Example:

```
<material id="1" type="solid mixture">
  <solid type="neo-Hookean">
    <E>1000.0</E>
    <v>0.45</v>
  </solid>
  <solid type="ellipsoidal fiber distribution">
    <mat_axis type="vector">
      <a>0.8660254, 0.5, 0</a>
      <d>0,0,1</d>
    </mat_axis>
    <ksi>5, 1, 1</ksi>
    <beta>2.5, 3, 3</beta>
  </solid>
  <solid type="ellipsoidal fiber distribution">
    <mat_axis type="vector">
      <a>0.8660254,-0.5, 0</a>
      <d>0,0,1</d>
    </mat_axis>
    <ksi>5, 1, 1</ksi>
    <beta>2.5, 3, 3</beta>
  </solid>
</material>
```

4.1.4.27 Spherical Fiber Distribution

The material type for a spherical (isotropic) continuous fiber distribution is "*spherical fiber distribution*". Since fibers can only sustain tension, this material is not stable on its own. It must be combined with a stable compressible material that acts as a ground matrix, using a "*solid mixture*" container as described in Section 4.1.4.26. The following material parameters need to be defined:

<alpha>	parameter α	[ ]
<beta>	parameter β	[ ]
<ksi>	parameters ξ	[**P**]

The Cauchy stress for this fibrous material is given by [52, 4, 6]:

$$\boldsymbol{\sigma} = \int_0^{2\pi} \int_0^{\pi} H\left(I_n - 1\right) \sigma_n\left(\mathbf{n}\right) \sin\varphi \, d\varphi \, d\theta \, .$$

Here, $I_n = \lambda_n^2 = \mathbf{N} \cdot \mathbf{C} \cdot \mathbf{N}$ is the square of the fiber stretch λ_n, $\mathbf{N}$ is the unit vector along the fiber direction, in the reference configuration, which in spherical angles is directed along (θ, φ), $\mathbf{n} = \mathbf{F} \cdot \mathbf{N}/\lambda_n$, and $H\left(.\right)$ is the unit step function that enforces the tension-only contribution.

The fiber stress is determined from a fiber strain energy function,

$$\sigma_n = \frac{2I_n}{J} \frac{\partial \Psi}{\partial I_n} \mathbf{n} \otimes \mathbf{n} \, ,$$

where in this material, the fiber strain energy density is given by

$$\Psi = \frac{\xi}{\alpha} \left(\exp\left[\alpha \left(I_n - 1\right)^{\beta} \right] - 1 \right) \, ,$$

where $\xi > 0$, $\alpha \geqslant 0$, and $\beta \geqslant 2$.

Note: In the limit when $\alpha \to 0$, this expressions produces a power law,

$$\lim_{\alpha \to 0} \Psi = \xi \left(I_n - 1\right)^{\beta}$$

Note: When $\beta > 2$, the fiber modulus is zero at the strain origin ($I_n = 1$). Therefore, use $\beta > 2$ when a smooth transition in the stress is desired from compression to tension.

Example:

```
<material id="1" type="solid mixture">
  <solid type="neo-Hookean">
    <E>1000.0</E>
    <v>0.45</v>
  </solid>
  <solid type="spherical fiber distribution">
    <ksi>10</ksi>
    <alpha>0</alpha>
    <beta>2.5</beta>
  </solid>
</material>
```

4.1.4.28 Spherical Fiber Distribution from Solid-Bound Molecule

The material type for a spherical (isotropic) continuous fiber distribution with fiber modulus dependent on solid-bound molecule content is "*spherical fiber distribution sbm*". Since fibers can only sustain tension, this material is not stable on its own. It must be combined with a stable compressible material that acts as a ground matrix, using a "*solid mixture*" container as described in Section 4.1.4.26. The following material parameters need to be defined:

<alpha>	parameter α	[ ]
<beta>	parameter β	[ ]
<ksi0>	fiber modulus ξ_0	[**P**]
<gamma>	fiber modulus exponent γ	[ ]
<rho0>	fiber mass density ρ_0	[**M**/**L**^{3}]
sbm	index of solid-bound molecule σ	[ ]

The Cauchy stress for this fibrous material is given by [52, 4, 6]:

$$\boldsymbol{\sigma} = \int_0^{2\pi} \int_0^{\pi} H\left(I_n - 1\right) \sigma_n\left(\mathbf{n}\right) \sin\varphi \, d\varphi \, d\theta \, .$$

Here, $I_n = \lambda_n^2 = \mathbf{N} \cdot \mathbf{C} \cdot \mathbf{N}$ is the square of the fiber stretch λ_n, $\mathbf{N}$ is the unit vector along the fiber direction, in the reference configuration, which in spherical angles is directed along (θ, φ), $\mathbf{n} = \mathbf{F} \cdot \mathbf{N}/\lambda_n$, and $H\left(.\right)$ is the unit step function that enforces the tension-only contribution.

The fiber stress is determined from a fiber strain energy function,

$$\sigma_n = \frac{2I_n}{J} \frac{\partial \Psi}{\partial I_n} \mathbf{n} \otimes \mathbf{n} \, ,$$

where in this material, the fiber strain energy density is given by

$$\Psi = \frac{\xi}{\alpha} \left(\exp\left[\alpha \left(I_n - 1\right)^{\beta} \right] - 1 \right) \, ,$$

where $\xi > 0$, $\alpha \geqslant 0$, and $\beta \geqslant 2$. The fiber modulus is dependent on the solid-bound molecule referential density ρ_r^{σ} according to the power law relation

$$\xi = \xi_0 \left(\frac{\rho_r^{\sigma}}{\rho_0} \right)^{\gamma} \, ,$$

where ρ_0 is the density at which $\xi = \xi_0$.

This type of material references a solid-bound molecule that belongs to a multiphasic mixture. Therefore this material may only be used as the solid (or a component of the solid) in a multiphasic mixture (Section 4.12). The solid-bound molecule must be defined in the <Globals> section (Section 3.4.3) and must be included in the multiphasic mixture using a <solid_bound> tag. The parameter *sbm* must refer to the global index of that solid-bound molecule. The value of ρ_r^{σ} is specified within the <solid_bound> tag. If a chemical reaction is defined within that multiphasic mixture that alters the value of ρ_r^{σ}, lower and upper bounds may be specified for this referential density within the <solid_bound> tag to prevent ξ from reducing to zero or achieving excessively elevated values.

Note: In the limit when $\alpha \to 0$, the expression for Ψ produces a power law,

$$\lim_{\alpha \to 0} \Psi = \xi \left(I_n - 1\right)^{\beta}$$

Note: When $\beta > 2$, the fiber modulus is zero at the strain origin ($I_n = 1$). Therefore, use $\beta > 2$ when a smooth transition in the stress is desired from compression to tension.

Example:

```
<solid type="solid mixture">
  <solid type="neo-Hookean">
    <E>1000.0</E>
    <v>0.45</v>
  </solid>
  <solid type="spherical fiber distribution sbm">
    <alpha>0</alpha>
    <beta>2.5</beta>
    <ksi0>10</ksi0>
    <gamma>2</gamma>
    <rho0>1</rho0>
    <sbm>1</sbm>
  </solid>
</solid>
```

4.1.4.29 Coupled Transversely Isotropic Mooney-Rivlin

This material describes a transversely isotropic Mooney-Rivlin material using a fully-coupled formulation. It is define through the *coupled trans-iso Mooney-Rivlin* material type. The following material parameters must be defined.

c1	Mooney-Rivlin c_1 parameter.	[**P**]
c2	Mooney-Rivlin c_2 parameter.	[**P**]
c3	exponential multiplier	[**P**]
c4	fiber scale factor	[ ]
c5	fiber modulus in linear region	[**P**]
lam_max	maximum fiber straightening stretch	[ ]
k	bulk-like modulus	[**P**]

The strain-energy function for this constitutive model is defined by

$$W = c_1 (I_1 - 3) + c_2 (I_2 - 3) - 2 (c_1 + 2c_2) \ln J + F(\lambda) + U(J)$$

The first three terms define the coupled Mooney-Rivlin matrix response. The third term is the fiber response which is a function of the fiber stretch λ and is defined as in [83]. For U, the following form is chosen in FEBio.

$$U(J) = \frac{1}{2} k (\ln J)^2$$

where $J = \det \mathbf{F}$is the Jacobian of the deformation.

Example:

```
<material id="1" type="coupled trans-iso Mooney-Rivlin">
  <c1>1</c1>
  <c2>0.1</c2>
  <c3>1</c3>
  <c4>1</c4>
  <c5>1.34</c5>
  <lam_max>1.3</lam_max>
  <k>100</k>
</material>
```

4.1.4.30 Coupled Transversely Isotropic Veronda-Westmann

This material describes a transversely isotropic Veronda-Westmann material using a fully-coupled formulation. It is define through the *coupled trans-iso Veronda-Westmann* material type. The following material parameters must be defined.

c1	Veronda-Westmann c_1 parameter.	[**P**]
c2	Veronda-Westmann c_2 parameter.	[ ]
c3	exponential multiplier	[**P**]
c4	fiber scale factor	[ ]
c5	fiber modulus in linear region	[**P**]
lam_max	maximum fiber straightening stretch	[ ]
k	bulk-like modulus	[**P**]

The strain-energy function for this constitutive model is defined by

$$W = c_1 \left(e^{c_2 (I_1 - 3)} - 1 \right) - \frac{1}{2} c_1 c_2 \left(I_2 - 3 \right) + F \left(\lambda \right) + U \left(J \right) .$$

The first two terms define the coupled Veronda-Westmann matrix response. The third term is the fiber response which is a function of the fiber stretch λ and is defined as in [83]. For U, the following form is chosen in FEBio.

$$U \left(J \right) = \frac{1}{2} k \left(\ln J \right)^2$$

where $J = \det \mathbf{F}$is the Jacobian of the deformation.

Example:

```
<material id="1" type="coupled trans-iso Veronda-Westmann">
  <c1>1</c1>
  <c2>0.1</c2>
  <c3>1</c3>
  <c4>1</c4>
  <c5>1.34</c5>
  <lam_max>1.3</lam_max>
  <k>100</k>
</material>
```

4.1.4.31 Large Poisson's Ratio Ligament

This material describes a coupled, transversely isotropic material that will conform to a particular Poisson's ratio when stretched along the fiber direction [77]. The following material parameters must be defined:

c1	Fiber coefficient c_1
c2	Fiber coefficient c_2
mu	Matrix coefficient μ
v0	Poisson's ratio parameter v_0
m	Poisson's ratio parameter m
k	Volumetric penalty coefficient κ

The strain energy function for this constitutive model is a three part expression:

$$W = W_{\text{fiber}} + W_{\text{matrix}} + W_{\text{vol}},$$

where:

$$W_{\text{fiber}} = \frac{1}{2}\frac{c_1}{c_2}\left(e^{c_2(\lambda-1)^2-1}\right),$$
$$W_{\text{matrix}} = \frac{\mu}{2}(I_1-3) - \mu\ln\left(\sqrt{I_3}\right),$$
$$W_{\text{vol}} = \frac{\kappa}{2}\left(\ln\left(\frac{I_5 - I_1 I_4 + I_2}{I_4^{2(m-v_0)}e^{-4m(\lambda-1)}}\right)\right)^2.$$

In the equations above, λ is the stretch ratio of the material along the fiber direction. The desired Poisson's ratio must first be selected based on available data for uniaxial tension along the fiber direction. The function with which to fit the Poisson's ratio data is:

$$v = -\frac{\lambda^{m-v_0}e^{-m(\lambda-1)} - 1}{\lambda - 1}.$$

The volumetric penalty coefficient κ must be selected to be large enough to enforce the Poisson's function above. If this material is to be used in a biphasic representation, κ must be selected based on experimental stress-relaxation data, since κ has an effect on the biphasic behavior of the material. Once κ, v_0, and m are chosen, c_1, c_2 and μ should be selected by fitting the stress-strain behavior of the material to experimental data. The Cauchy stress of the material is given by:

$$\boldsymbol{\sigma} = \frac{2}{J}\left((W_1 + I_1 W_2)\,\mathbf{B} - W_2\mathbf{B}^2 + W_3\,(I_3\mathbf{1}) + W_4 I_4\,(\mathbf{a}\otimes\mathbf{a}) + W_5 I_4\,(\mathbf{a}\otimes\mathbf{a}\cdot\mathbf{B} + \mathbf{B}\cdot\mathbf{a}\otimes\mathbf{a})\right)$$

where J is the jacobian of the deformation gradient $\mathbf{F}$, $\mathbf{B} = \mathbf{F}\cdot\mathbf{F}^T$ is the left Cauchy-Green deformation tensor, $\mathbf{1}$ is the 3×3 identity tensor**,** $\mathbf{a}$ is the fiber orientation vector in the deformed configuration.

Example:

```
<material id="1" name="Material1" type="PRLig">
  <c1>90</c1>
  <c2>160</c2>
  <mu>0.025</mu>
```

```
    <v0>5.85</v0>
    <m>-100</m>
    <k>1.55</k>
  </material>
```

4.1.4.32 Shenoy-Wang material

This material implements the constitutive model by Wang et al. (Biophysical Journal, 107, 2014, pp:2592 - 2603), which proposes a mechanism for long-range force transmission in fibrous matrices enabled by tension-driven alignment of fibers.

It defines the following parameters.

mu	matrix shear modulus μ
k	matrix bulk modulus k
Ef	Fiber stiffness modulus E_f
lam_c	transition point λ_c
lam_t	transition interval size λ_t
n	strain hardening exponent *n*
m	strain hardening exponent *m*

The strain-energy density function is given by,

$$W = W_{\mathsf{b}} + W_{\mathsf{f}},$$

where:

$$W_{\mathsf{b}} = \frac{\mu}{2}\left(\overline{I}_1 - 3\right) + \frac{\kappa}{2}\left(J-1\right)^2,$$
$$W_{\mathsf{f}} = \overset{[}{a} = 1]3\sum f\left(\lambda_a\right),$$

and $f$is defined via its derivative,

$$\frac{\partial f}{\partial \lambda_a}\left(\lambda_a\right) = \begin{cases} 0, & \lambda_a < \lambda_1 \\ \frac{E_f\left(\frac{\lambda_a-\lambda_1}{\lambda_2-\lambda_1}\right)^n(\lambda_a-\lambda_1)}{n+1}, & \lambda_1 \le \lambda_a < \lambda_2 \\ E_f\left[\frac{\lambda_2-\lambda_1}{n+1} + \frac{(1+\lambda_a-\lambda_2)^{m+1}-1}{m+1}\right], & \lambda_a \ge \lambda_2 \end{cases}$$

Finally, the parameters $\lambda_1$and $\lambda_2$are given as follows.

$$\lambda_1 = \lambda_c - \lambda_t/2$$
$$\lambda_2 = \lambda_c + \lambda_t/2$$

Example:

```
<material id="1" name="Material1" type="Shenoy">
  <density>1</density>
  <mu>0.7692</mu>
  <k>1.667</k>
  <Ef>134.6</Ef>
  <lam_c>1.02</lam_c>
  <lam_t>0.255</lam_t>
  <n>5</n>
  <m>30</m>
</material>
```

4.2 Fibers

Fiber materials are used to model one-dimensional structures oriented along a unit vector. The associated strain energy density is a function of the normal strain along that vector. These fiber models are constructed such that the strain energy is non-zero only when the fiber is in tension, under the idealized assumption that fibers do not sustain any load in compression. This assumption produces an inherent instability in the material response of fibers, therefore such models must be combined with elastic materials that can sustain compression and tension, thus serving as models of a ground matrix.

FEBio accommodates fiber models that can be combined with unconstrained or uncoupled models of the ground matrix. Historically, unconstrained models of a ground matrix have been combined with unconstrained fiber models, whereas uncoupled models of a ground matrix have been combined with uncoupled fiber models. The manual sections presented below follow this conventional approach.

However, it should be noted that some authors have expressed concerns about using uncoupled fiber formulations in fiber-matrix material models [68, 41, 40], due to two fundamental concerns: When highly nonlinear fibers become much stiffer than the ground matrix upon loading, it may become difficult to enforce an isochoric deformation, as would be expected in an uncoupled formulation. Furthermore, in an uncoupled formulation, the strain is split into its deviatoric (isochoric) and volumetric parts and the fiber strain is evaluated only from the deviatoric part. This deviatoric fiber strain is not the true fiber strain, yet it is used to determine whether the fiber is in tension. These factors may result in non-physical deformations, as described by Helfenstein et al. [41].

The solution advocated by these investigators has been to use an uncoupled formulation for the ground matrix only, while the fiber models should remain unconstrained [68, 41, 40]. This can be done in FEBio by using a "*uncoupled solid mixture*" container as described in Section 4.1.2.16, where the material representing the ground matrix is taken from the list of uncoupled materials in Section 4.1.2, whereas fiber models are taken from the list of unconstrained models in Section 4.2.1 or Section 4.3.1.

4.2.1 Unconstrained Fiber Models

Since fibers can only sustain tension, the materials listed here are not stable on their own. They must be combined with a stable material that acts as a ground matrix, using a "*solid mixture*" container as described in Section 4.1.4.26, within a <solid> tag. These fiber models may also be used in continuous fiber distribution models as described in Section 4.3.1, within a <fibers> tag.

4.2.1.1 Fiber with Exponential-Power Law

The material type for a single fiber with an exponential-power law is "*fiber-exp-pow*". Since fibers can only sustain tension, this material is not stable on its own. It must be combined with a stable compressible material that acts as a ground matrix, using a "*solid mixture*" container as described in Section 4.1.4.26. The following material parameters need to be defined:

<ksi>	ξ, representing a measure of the fiber modulus	[**P**]
<alpha>	α, coefficient of exponential argument	[ ]
<beta>	β, power of exponential argument	[ ]
<lam0>	λ_0, stretch threshold for tensile response	[ ]

The fiber is oriented along the unit vector $\mathbf{e}_1$, where $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ are orthonormal basis vectors representing the local element coordinate system when specified (Section 4.1.1), or else the global Cartesian coordinate system. The Cauchy stress for this fibrous material is given by

$$\boldsymbol{\sigma} = H\left(I_n - I_0\right) \frac{2I_n}{J} \frac{\partial \Psi}{\partial I_n} \mathbf{n} \otimes \mathbf{n},$$

where $I_n = \lambda_n^2 = \mathbf{n}_r \cdot \mathbf{C} \cdot \mathbf{n}_r$ is the square of the fiber stretch, $\mathbf{n} = \mathbf{F} \cdot \mathbf{n}_r / \lambda_n$, $I_0 = \lambda_0^2$ is the square of the stretch threshold for the tensile response ($\lambda_0 = 1$ by default) and $H\left(.\right)$ is the unit step function that enforces the tension-only contribution. The fiber strain energy density is given by

$$\Psi = \frac{\xi}{\alpha\beta} \left(\exp\left[\alpha \left(I_n - I_0\right)^\beta \right] - 1 \right),$$

where $\xi > 0$, $\alpha \geqslant 0$, and $\beta \geqslant 2$.

Note: In the limit when $\alpha \to 0$, this expressions produces a power law,

$$\lim_{\alpha \to 0} \Psi = \frac{\xi}{\beta} \left(I_n - I_0\right)^\beta$$

Note: When $\beta > 2$, the fiber modulus is zero at the strain origin ($I_n = I_0$). Therefore, use $\beta > 2$ when a smooth transition in the stress is desired from compression to tension.

Example:

Single fiber oriented along $\mathbf{e}_1$, embedded in a neo-Hookean ground matrix.

```
<material id="1" type="solid mixture">
  <mat_axis type="local">0,0,0</mat_axis>
  <solid type="neo-Hookean">
    <E>1000.0</E>
    <v>0.45</v>
  </solid>
  <solid type="fiber-exp-pow">
    <ksi>5</ksi>
    <alpha>20</alpha>
    <beta>3</beta>
    <mat_axis type="angles">
        <theta>0</theta>
        <phi>90</phi>
    </mat_axis>
  </solid>
</material>
```

Example:

Two fibers in the plane orthogonal to $\mathbf{e}_1$, oriented at ±25 degrees relative to $\mathbf{e}_3$, embedded in a neo-Hookean ground matrix.

```
<material id="1" type="solid mixture">
  <mat_axis type="local">0,0,0</mat_axis>
  <solid type="neo-Hookean">
    <E>1000.0</E>
    <v>0.45</v>
  </solid>
  <solid type="fiber-exp-pow">
    <ksi>5</ksi>
    <alpha>20</alpha>
    <beta>3</beta>
    <fiber type="angles">
      <theta>90</theta>
      <phi>25</phi>
    </fiber>
  </solid>
  <solid type="fiber-exp-pow">
    <ksi>5</ksi>
    <alpha>20</alpha>
    <beta>3</beta>
    <fiber type="angles">
      <theta>-90</theta>
      <phi>25</phi>
    </fiber>
  </solid>
</material>
```

4.2.1.2 Fiber with Neo-Hookean Law

This material type is "*fiber-NH*". The following material parameters need to be defined:

<mu>	μ, representing a measure of the fiber modulus	**[P]**

The fiber strain energy density is given by

$$\Psi_n\left(I_n\right) = \frac{\mu}{4}\left(I_n - 1\right)^2 \ ,$$

where $\mu > 0$.

Example:

Fiber model as specified in a continuous fiber distribution (Section 4.3.1)

```
<fibers type="fiber-NH">
  <mu>1</mu>
</fibers>
```

4.2.1.3 Fiber with Natural Neo-Hookean Law

This material type is "*fiber-natural-NH*". The following material parameters need to be defined:

<ksi>	ξ, fiber modulus	[**P**]
<lam0>	λ_0, stretch threshold for tensile response	[ ]

The fiber strain energy density is given by

$$\Psi_n(\lambda_n) = \frac{\xi}{2} H\left(\ln\frac{\lambda_n}{\lambda_0}\right)\left(\ln\frac{\lambda_n}{\lambda_0}\right)^2,$$

where $\xi > 0$ and $\lambda_0 \geq 1$. The tensile response engages at the tensile stretch ratio λ_0 ($\lambda_0 = 1$ by default).The natural (logarithmic or Hencky) strain along the fiber is $\ln\lambda_n$, whereas $\ln\frac{\lambda_n}{\lambda_0} = \ln\lambda_n - \ln\lambda_0$ is the natural strain relative to the tensile stretch threshold. This model produces a stress response which varies linearly with the natural strain.

Example:

Fiber model as specified in a solid mixture (Section 4.1.4.26)

```
<material id="1" name="Material1" type="solid mixture">
  <density>1</density>
  <solid type="neo-Hookean">
    <E>0.13</E>
    <v>0.3</v>
  </solid>
  <solid type="fiber-natural-NH">
    <ksi>1</ksi>
    <lam0>1.25</lam0>
    <fiber type="vector">0,0,1</fiber>
  </solid>
</material>
```

4.2.1.4 Fiber with Toe-Linear Response

This material type is "*fiber-pow-linear*". The following material parameters need to be defined:

<E>	E, the fiber modulus in the linear range ($E \geqslant 0$)	[**P**]
<beta>	β, the power-law exponent in the toe region ($\beta \geqslant 2$)	[ ]
<lam0>	λ_0, the stretch ratio when the toe region transitions to the linear region ($\lambda_0 > 1$)	[ ]

The fiber strain energy density is given by

$$\Psi_n(I_n) = \begin{cases} 0 & I_n < 1 \\ \frac{\xi}{\beta}(I_n - 1)^\beta & 1 \leqslant I_n \leqslant I_0 \\ B(I_n - I_0) - E\left(I_n^{1/2} - I_0^{1/2}\right) + \frac{\xi}{\beta}(I_0 - 1)^\beta & I_0 < I_n \end{cases},$$

where $I_0 = \lambda_0^2$,

$$\xi = \frac{E}{4(\beta - 1)} I_0^{-3/2}(I_0 - 1)^{2-\beta}, \; B = \xi(I_0 - 1)^{\beta - 1} + \frac{E}{2} I_0^{-1/2}$$

For this material type, the fiber elasticity at the strain origin reduces to zero unless $\beta = 2$.

Example:
Fiber model as specified in a solid mixture (Section 4.1.4.26)

```
<solid type="fiber-pow-linear">
  <fiber type="angles">
    <theta>20</center>
    <phi>90</phi>
  </fiber>
  <E>1</E>
  <beta>2.5</beta>
  <lam0>1.06</lam0>
</solid>
```

4.2.1.5 Fiber with Exp-Pow-Linear Response

This material type is "*fiber-exp-pow-linear*". The following material parameters need to be defined:

<E>	E, the fiber modulus in the linear range ($E \geqslant 0$)	[**P**]
<alpha>	α, coefficient of exponential argument	[ ]
<beta>	β, the power-law exponent in the toe region ($\beta \geqslant 2$)	[ ]
<lam0>	λ_0, the stretch ratio when the toe region transitions to the linear region ($\lambda_0 > 1$)	[ ]

The fiber strain energy density is given by

$$\Psi_n = \begin{cases} 0 & I_n < 1 \\ \frac{\xi}{\alpha\beta}\left(\exp\left[\alpha\left(I_n-1\right)^{\beta}\right]-1\right) & 1 \leq I_n \leq I_0 \\ B\left(I_n - I_0\right) - E\left(I_n^{1/2} - I_0^{1/2}\right) + \frac{\xi}{\alpha\beta}\left(\exp\left[\alpha\left(I_0-1\right)^{\beta}\right]-1\right) & I_n > I_0 \end{cases}$$

where $I_0 = \lambda_0^2$,

$$\xi = \frac{E\left(I_0-1\right)^{2-\beta}\exp\left[-\alpha\left(I_0-1\right)^{\beta}\right]}{4I_0^{3/2}\left(\beta-1+\alpha\beta\left(I_0-1\right)^{\beta}\right)}$$

and

$$B = E\frac{2I_0\left(\beta-\frac{1}{2}+\alpha\beta\left(I_0-1\right)^{\beta}\right)-1}{4I_0^{3/2}\left(\beta-1+\alpha\beta\left(I_0-1\right)^{\beta}\right)}$$

For this material type, the fiber elasticity at the strain origin reduces to zero unless $\beta = 2$. This model reduces to "*fiber-pow-linear*" when $\alpha = 0$. Alternatively, in the limit when $I_0 = 1$, the above parameters reduce to $\xi = 0$ and $B = E/2$ and the strain energy density takes the quadratic form

$$\Psi_n = \begin{cases} 0 & I_n < 1 \\ \frac{E}{2}\left(I_n^{1/2}-1\right)^2 & I_n > 1 \end{cases}$$

where $I_n^{1/2} = \lambda_n$ is the stretch ratio along the fiber.

Example:

Fiber model as specified in a solid mixture (Section 4.1.4.26)

```
<solid type="fiber-exp-pow-linear">
 <E>1080</E>
 <alpha>1400</alpha>
 <beta>2.73</beta>
 <lam0>1.012</lam0>
 <fiber type="vector">0,0,1</fiber>
</solid>
```

4.2.1.6 Fiber Exp-Linear

This material type is "*fiber-exp-linear*". This constitutive fiber model has an initial exponential rise and then grows linearly after a user-specified stretch transition point. This fiber material is based on the trans-iso Mooney-Rivlin model introduced in [83]. This material by itself is not stable, so it is recommend to use it as part of a solid mixture.

The strain energy is as follows:

$$F_2(\lambda) = \begin{cases} 0 & \lambda \leqslant 1 \\ C_3\left(e^{-C_4}\left(\mathrm{Ei}\left(C_4\lambda\right) - \mathrm{Ei}\left(C_4\right)\right) - \ln\lambda\right) & 1 < \lambda < \lambda_m \\ C_5\left(\lambda - 1\right) + C_6 \ln\lambda & \lambda \geqslant \lambda_m \end{cases}$$

where $\mathrm{Ei}\left(\cdot\right)$ is the exponential integral function. The resulting fiber stress is evaluated from

$$\lambda\frac{\partial F_2}{\partial\lambda} = \begin{cases} 0 & \tilde{\lambda} \leqslant 1 \\ C_3\left(e^{C_4(\lambda-1)} - 1\right) & 1 < \tilde{\lambda} < \lambda_m \\ C_5\lambda + C_6 & \tilde{\lambda} \geqslant \lambda_m \end{cases}.$$

Here, λ_m is the stretch at which the fibers are straightened, C_3scales the exponential stresses, C_4 is the rate of uncrimping of the fibers, and $C_5$is the modulus of the straightened fibers. C_6 is determined from the requirement that the stress is continuous at λ_m.

<c3>	Exponential stress coefficient	[**P**]
<c4>	Fiber uncrimping coefficient	[ ]
<c5>	Modulus of straightened fibers	[**P**]
<lambda>	Fiber stretch for straightened fibers	[ ]

Example:

```
<material id="1" name="Material" type="solid mixture">
  <solid type="neo-Hookean">
    <E>1e6</E>
    <v>0.32</v>
  </solid>
  <solid type="fiber-exp-linear">
    <c3>1</c3>
    <c4>1</c4>
    <c5>1</c5>
    <lambda>1.05</lambda>
    <fiber type="const">1,0,0</fiber>
  </solid>
</material>
```

4.2.1.7 Fiber as Entropy Chain

This material type is "*fiber-entropy-chain*". It was proposed by [69]. This fiber model is based on statistical mechanics to reflect the entropy-driven nature of a biological fiber. The model is derived from the freely-jointed-chain mechanism. Its strain energy is directly related to the entropic change of the chains in the material, given by

$$\Psi_n\left(I_n\right) = \xi N\left(\sqrt{\frac{I_n}{N}}\beta + \ln\frac{\beta}{\sinh\beta}\right) - \frac{\xi\sqrt{N}}{2}\beta_0 I_n - \alpha_{00}$$

where $I_n = \mathbf{n}_r \cdot \mathbf{C} \cdot \mathbf{n}_r$ is the square of the stretch ratio along the referential fiber direction $\mathbf{n}_r$. The inverse Langevin equation relates the parameter β to I_n and N according to $\beta = \mathcal{L}^{-1}\left[\sqrt{\frac{I_n}{N}}\right]$. Here, β_0 is the value of β when $I_n = 1$, andα_{00} is needed to produce a state of zero energy at $I_n = 1$. The parameter $\xi = nk\Theta$ is the initial fiber stiffness, with n, k, and Θ respectively representing the number of chains per unit volume, Boltzmann's constant, and the absolute temperature. The parameter $N = \zeta^2$ is the number of chain segments, and ζ is the locking stretch, representing the extensibility of the fiber. The Langevin function is given by $\mathcal{L}\left(x\right) = \coth x - \frac{1}{x}$.

When evaluating the inverse Langevin equation, a Taylor series expansion is applied for computational stability. The parameter n_{term} is used to control the number of terms used to evaluate the equation; it must be an integer between 3 and 30.
The following material parameters must be defined:

<ksi>	Initial modulus	[**P**]
<N>	Square of locking stretch	[ ]
<n_term>	Number of Taylor series terms	
<mu>	Shear modulus fir fiber-matrix interaction	[**P**]

Example:

```
<material id="1" name="Soft Tissue" type="solid mixture">
<solid type="Arruda-Boyce unconstrained">
  <N>2</N>
  <ksi>1</ksi>
  <n_term>30</n_term>
  <kappa>1</kappa>
        </solid>
        <solid type="continuous fiber distribution">
          <mat_axis type="angles">
            <theta>0</theta>
            <phi>0</phi>
          </mat_axis>
          <fibers type="fiber-entropy-chain">
            <ksi>1</ksi>
            <N>2</N>
            <n_term>30</n_term>
          </fibers>
          <distribution type="von-Mises-3d">
```

```
      <b>0.3</b>
    </distribution>
    <scheme type="fibers-3d-fei">
      <resolution>196</resolution>
    </scheme>
  </solid>
</material>
```

4.2.2 Uncoupled Fiber Models

Since fibers can only sustain tension, the materials listed here are not stable on their own. They must be combined with a stable material that acts as a ground matrix, using a "*uncoupled solid mixture*" container as described in Section 4.1.2.16, within a <solid> tag. These fiber models may also be used in continuous fiber distribution models as described in Section 4.3.2, using a <fibers> tag.

4.2.2.1 Fiber with Exponential-Power Law, Uncoupled Formulation

The material type for a single fiber with an exponential-power law, in an uncoupled strain energy formulation, is "*fiber-exp-pow-uncoupled*". Since fibers can only sustain tension, this material is not stable on its own. It must be combined with a stable uncoupled material that acts as a ground matrix, using a "*uncoupled solid mixture*" container as described in Section 4.1.2.16. The following material parameters need to be defined:

<ksi>	ξ, representing a measure of the fiber modulus	[**P**]
<alpha>	α, coefficient of exponential argument	[ ]
<beta>	β, power of exponential argument	[ ]

The fiber is oriented along the unit vector $\mathbf{e}_1$, where $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ are orthonormal basis vectors representing the local element coordinate system when specified (Section 4.1.1), or else the global Cartesian coordinate system. The stress $\tilde{\boldsymbol{\sigma}}$ for this fibrous material is given by

$$\tilde{\boldsymbol{\sigma}} = H\left(\tilde{I}_n - 1\right) \frac{2\tilde{I}_n}{J} \frac{\partial \tilde{\Psi}}{\partial \tilde{I}_n} \mathbf{n} \otimes \mathbf{n},$$

where $\tilde{I}_n = \tilde{\lambda}_n^2 = \mathbf{n}_r \cdot \tilde{\mathbf{C}} \cdot \mathbf{n}_r$ is the square of the fiber stretch, $\mathbf{n} = \tilde{\mathbf{F}} \cdot \mathbf{n}_r / \tilde{\lambda}_n$, and $H(.)$ is the unit step function that enforces the tension-only contribution.. The fiber strain energy density is given by

$$\tilde{\Psi} = \frac{\xi}{\alpha\beta}\left(\exp\left[\alpha\left(\tilde{I}_n - 1\right)^{\beta}\right] - 1\right),$$

where $\xi > 0$, $\alpha \geqslant 0$, and $\beta \geqslant 2$.

Note: In the limit when $\alpha \to 0$, this expressions produces a power law,

$$\lim_{\alpha \to 0} \tilde{\Psi} = \frac{\xi}{\beta}\left(\tilde{I}_n - 1\right)^{\beta}.$$

Note: When $\beta > 2$, the fiber modulus is zero at the strain origin ($\tilde{I}_n = 1$). Therefore, use $\beta > 2$ when a smooth transition in the stress is desired from compression to tension.

Example:

Single fiber oriented along $\mathbf{e}_1$, embedded in a Mooney-Rivlin ground matrix.

```
<material id="1" type="uncoupled solid mixture">
  <mat_axis type="local">0,0,0</mat_axis>
  <k>10e3</k>
  <solid type="Mooney-Rivlin">
    <c1>10.0</c1>
    <c2>0</c2>
  </solid>
  <solid type="fiber-exp-pow-uncoupled">
    <ksi>5</ksi>
    <alpha>20</alpha>
    <beta>3</beta>
    <mat_axis type="angles">
        <theta>0</theta>
        <phi>90</phi>
```

```
    </mat_axis>
  </solid>
</material>
```

Example:

Two fibers in the plane orthogonal to $\mathbf{e}_1$, oriented at ±25 degrees relative to $\mathbf{e}_3$, embedded in a Mooney-Rivlin ground matrix.

```
<material id="1" type="uncoupled solid mixture">
  <mat_axis type="local">0,0,0</mat_axis>
  <k>10e3</k>
  <solid type="Mooney-Rivlin">
    <c1>10.0</c1>
    <c2>0</c2>
  </solid>
  <solid type="fiber-exp-pow-uncoupled">
    <ksi>5</ksi>
    <alpha>20</alpha>
    <beta>3</beta>
    <mat_axis type="angles">
        <theta>90</theta>
        <phi>25</phi>
    </mat_axis>
  </solid>
  <solid type="fiber-exp-pow-uncoupled">
    <ksi>5</ksi>
    <alpha>20</alpha>
    <beta>3</beta>
    <mat_axis type="angles">
        <theta>-90</theta>
        <phi>25</phi>
    </mat_axis>
  </solid>
</material>
```

4.2.2.2 Fiber Kiousis Uncoupled

This material type is "*fiber-Kiousis-uncoupled*". It is based on the material model proposed by Kiousis et al. [48, 49] for modeling the balloon in a balloon-stent
+system. The following material parameters need to be defined:

<d1>	d_1, represents a measure of the fiber modulus	[**P**]
<d2>	d_2, square of the stretch ratio when the fiber engages	
<n>	n, power exponent	

The fiber strain energy density is given by

$$\tilde{\Psi}_n\left(I_n\right) = \begin{cases} \frac{d_1}{n}\left(\tilde{I}_n - d_2\right)^n & \tilde{I}_n \geq d_2 \\ 0 & \tilde{I}_n < d_2 \end{cases},$$

where $\tilde{I}_n = \tilde{\lambda}_n^2 = \mathbf{n}_r \cdot \tilde{\mathbf{C}} \cdot \mathbf{n}_r$ is the square of the fiber stretch and $\mathbf{n}_r$ is the unit vector along the fiber direction in its reference configuration.

Example:
Fiber model as specified in a continuous fiber distribution (Section 4.3.2)

```
<fibers type="fiber-Kiousis-uncoupled">
  <d1>500</d1>
  <d2>2.25</d2>
  <n>3</n>
</fibers>
```

4.2.2.3 Fiber with Toe-Linear Response, Uncoupled Formulation

This material type is "*fiber-pow-linear-uncoupled*". The following material parameters need to be defined:

<E>	E, the fiber modulus in the linear range ($E \geqslant 0$)	[**P**]
<beta>	β, the power-law exponent in the toe region ($\beta \geqslant 2$)	[ ]
<lam0>	λ_0, the stretch ratio when the toe region transitions to the linear region ($\lambda_0 > 1$)	[ ]

The fiber strain energy density is given by

$$\tilde{\Psi}_n\left(\tilde{I}_n\right) = \begin{cases} 0 & \tilde{I}_n < 1 \\ \frac{\xi}{\beta}\left(\tilde{I}_n - 1\right)^{\beta} & 1 \leqslant \tilde{I}_n \leqslant I_0 \\ B\left(\tilde{I}_n - I_0\right) - E\left(\tilde{I}_n^{1/2} - I_0^{1/2}\right) + \frac{\xi}{\beta}\left(I_0 - 1\right)^{\beta} & I_0 < \tilde{I}_n \end{cases},$$

where $I_0 = \lambda_0^2$,

$$\xi = \frac{E}{4\left(\beta - 1\right)} I_0^{-3/2}\left(I_0 - 1\right)^{2-\beta},\ B = \xi\left(I_0 - 1\right)^{\beta-1} + \frac{E}{2} I_0^{-1/2}.$$

Example:

Fiber model as specified in a solid mixture (Section 4.1.2.16)

```
<solid type="fiber-pow-linear-uncoupled">
  <fiber type="angles">
    <theta>20</center>
    <phi>90</phi>
  </fiber>
  <E>1</E>
  <beta>2.5</beta>
  <lam0>1.06</lam0>
</solid>
```

4.2.2.4 Uncoupled Fiber Exp-Linear

This constitutive fiber model has an initial exponential rise and then grows linearly after a user-specified stretch transition point. This fiber material is based on the trans-iso Mooney-Rivlin model introduced in [83]. This material by itself is not stable, so it is recommend to use it as part of a solid mixture.

The strain energy is as follows:

$$F_2\left(\tilde{\lambda}\right) = \begin{cases} 0 & \tilde{\lambda} \leqslant 1 \\ C_3\left(e^{-C_4}\left(\mathrm{Ei}\left(C_4\tilde{\lambda}\right) - \mathrm{Ei}\left(C_4\right)\right) - \ln\tilde{\lambda}\right) & 1 < \tilde{\lambda} < \lambda_m \\ C_5\left(\tilde{\lambda} - 1\right) + C_6 \ln\tilde{\lambda} & \tilde{\lambda} \geqslant \lambda_m \end{cases}$$

where $\mathrm{Ei}\,(\cdot)$ is the exponential integral function. The resulting fiber stress is evaluated from

$$\tilde{\lambda}\frac{\partial F_2}{\partial \tilde{\lambda}} = \begin{cases} 0 & \tilde{\lambda} \leqslant 1 \\ C_3\left(e^{C_4(\tilde{\lambda}-1)}-1\right) & 1 < \tilde{\lambda} < \lambda_m \\ C_5\tilde{\lambda} + C_6 & \tilde{\lambda} \geqslant \lambda_m \end{cases} .$$

Here, λ_m is the stretch at which the fibers are straightened, C_3scales the exponential stresses, C_4 is the rate of uncrimping of the fibers, and $C_5$is the modulus of the straightened fibers. C_6 is determined from the requirement that the stress is continuous at λ_m.

<c3>	Exponential stress coefficient	[**P**]
<c4>	Fiber uncrimping coefficient	[ ]
<c5>	Modulus of straightened fibers	[**P**]
<k>	Bulk modulus	[**P**]
<lambda>	Fiber stretch for straightened fibers	[ ]
<fiber>	Fiber distribution option	

Example:

```
<material id="1" name="Material" type="uncoupled solid mixture">
  <k>1e4</k>
  <solid type="Mooney-Rivlin">
    <c1>1</c1>
    <c2>0</c2>
  </solid>
  <solid type="uncoupled fiber-exp-linear">
    <c3>1</c3>
    <c4>1</c4>
    <c5>1</c5>
    <lambda>1.05</lambda>
    <fiber type="const">1,0,0</fiber>
  </solid>
</material>
```

4.2.2.5 Uncoupled Fiber as Entropy Chain

This material type is "*uncoupled fiber-entropy-chain*". It was proposed by [69]. This fiber model is based on statistical mechanics to reflect the entropy-driven nature of a biological fiber. The model is derived from the freely-jointed-chain mechanism. Its strain energy is directly related to the entropic change of the chains in the material, given by

$$\Psi_n(\tilde{I}_n) = \xi N \left(\sqrt{\frac{\tilde{I}_n}{N}}\beta + \ln\frac{\beta}{\sinh\beta} \right) - \frac{\xi\sqrt{N}}{2}\beta_0\tilde{I}_n - \alpha_{00}$$

where $\tilde{I}_n = \mathbf{n}_r \cdot \tilde{\mathbf{C}} \cdot \mathbf{n}_r$ is the deviatoric measure of the stretch ratio along the referential fiber direction $\mathbf{n}_r$. The inverse Langevin equation relates the parameter β to I_n and N according to $\beta = \mathcal{L}^{-1}\left(\sqrt{\frac{I_n}{N}}\right)$. Here, β_0 is the value of β when $I_n = 1$, and α_{00} is needed to produce a state of zero energy at $I_n = 1$. The parameter $\xi = nk\Theta$ is the initial fiber stiffness, with n, k, and Θ respectively representing the number of chains per unit volume, Boltzmann's constant, and the absolute temperature. The parameter $N = \zeta^2$ is the number of chain segments, and ζ is the locking stretch, representing the extensibility of the fiber. The Langevin function is given by $\mathcal{L}(x) = \coth x - \frac{1}{x}$.

When evaluating the inverse Langevin equation, a Taylor series expansion is applied for computational stability. The parameter n_{term} is used to control the number of terms used to evaluate the equation; it must be an integer between 3 and 30.
The following material parameters must be defined:

<ksi>	Initial modulus	[**P**]
<N>	Square of locking stretch	[ ]
<n_term>	Number of Taylor series terms	

Example:

```
<material id="1" name="Soft Tissue" type="uncoupled solid mixture">
<k>1e4</k>
<solid type="Mooney-Rivlin">
  <c1>10.0</c1>
  <c2>0</c2>
</solid>
<solid type="uncoupled fiber-entropy-chain">
  <N>2.3</N>
  <ksi>1</ksi>
  <n_term>25</n_term>
  <fiber type="vector">0,0,1</fiber>
</solid>
</material>
```

4.3 Continuous Fiber Distribution

A continuous fiber distribution has a strain energy density that integrates the contributions from fiber bundles oriented along all directions emanating from a point in the continuum,

$$\Psi_r\left(\mathbf{C}\right) = \int_A H\left(I_n - 1\right) R\left(\mathbf{n}_r\right) \Psi_n\left(I_n\right)\, dA\,,$$

where $\mathbf{n}_r$ is the unit vector along the fiber orientation in the reference configuration, $I_n = \mathbf{n}_r \cdot \mathbf{C} \cdot \mathbf{n}_r$ is the normal component of $\mathbf{C}$ along $\mathbf{n}_r$ (also the square of the stretch ratio along that direction), and A represents the unit sphere (for 3D fiber distributions) or unit circle (for 2D fiber distributions) over which the integration is performed [**?**]. Thus, $\mathbf{n}_r$ spans all directions from the origin to points on the unit sphere or unit circle. In the integrand, Ψ_n represents the strain energy density of the fiber bundle oriented along $\mathbf{n}_r$; $H\left(\cdot\right)$ is the Heaviside unit step function that includes only fibers that are in tension; and $R\left(\mathbf{n}_r\right)$ is the fiber density distribution function that specifies the spatial fractional distribution of fibers. This function satisfies the constraint

$$\int_A R\left(\mathbf{n}_r\right)\, dA = 1\,.$$

Fiber constitutive models may be taken from the list given in Section 4.2.1.

For a material with an uncoupled strain energy density the corresponding expression is

$$\tilde{\Psi}_r\left(\tilde{\mathbf{C}}\right) = \int_A H\left(\tilde{I}_n - 1\right) R\left(\mathbf{n}_r\right) \tilde{\Psi}_n\left(\tilde{I}_n\right)\, dA\,,$$

where $\tilde{I}_n = \mathbf{n}_r \cdot \tilde{\mathbf{C}} \cdot \mathbf{n}_r$. In this case, fiber constitutive models may be taken from the list given in Section 4.2.2.

4.3.1 Unconstrained Continuous Fiber Distribution

The material type for an unconstrained continuous fiber distribution material is *"continuous fiber distribution"*. The following parameters must be defined:

<fibers>	Specification of the fiber material response $\Psi_n(I_n)$	
<distribution>	Specification of the fiber density distribution $R(\mathbf{n})$	
<scheme>	Numerical integration scheme	

The <fibers> tag encloses a description of the fiber constitutive relation and associated material properties, as may be selected from the list provided in Section 4.2.1. The <distribution> tag encloses a description of the fiber density distribution function, as may be selected from the list presented in Section 4.3.3. The <scheme> tag specifies the numerical integration scheme, as may be selected from the list presented in Section 4.3.4.

Example:

```
<material id="1" name="Material" type="solid mixture">
  <solid type="neo-Hookean">
    <density>1</density>
    <E>1</E>
    <v>0.3</v>
  </solid>
  <solid type="continuous fiber distribution">
    <fibers type="fiber-exponential-power-law">
      <alpha>0</alpha>
      <beta>2</beta>
      <ksi>1</ksi>
    </fibers>
    <distribution type="spherical">
    </distribution>
    <scheme type="fibers-3d-gkt">
      <nph>7</nph>
      <nth>11</nth>
    </scheme>
  </solid>
</material>
```

4.3.2 Uncoupled Continuous Fiber Distribution

The material type for an uncoupled continuous fiber distribution material is *"continuous fiber distribution uncoupled"*. The following parameters must be defined:

<fibers>	Specification of the fiber material response $\tilde{\Psi}_n\left(\tilde{I}_n\right)$
<distribution>	Specification of the fiber density distribution $R(\mathbf{n})$
<scheme>	Numerical integration scheme

The <fibers> tag encloses a description of the fiber constitutive relation and associated material properties, as may be selected from the list provided in Section 4.2.2. The <distribution> tag encloses a description of the fiber density distribution function, as may be selected from the list presented in Section 4.3.3. The <scheme> tag specifies the numerical integration scheme, as may be selected from the list presented in Section 4.3.4.

Example:

```
<material id="1" name="Material" type="uncoupled solid mixture">
  <solid type="Mooney-Rivlin">
    <density>1</density>
    <c1>1</c1>
    <c2>0.3</c2>
  </solid>
  <solid type="continuous fiber distribution uncoupled">
    <fibers type="fiber-exponential-power-law-uncoupled">
      <alpha>0</alpha>
      <beta>2</beta>
      <ksi>1</ksi>
    </fibers>
    <distribution type="spherical">
    </distribution>
    <scheme type="fibers-3d-gkt">
      <nph>7</nph>
      <nth>11</nth>
    </scheme>
  </solid>
</material>
```

4.3.3 Distribution

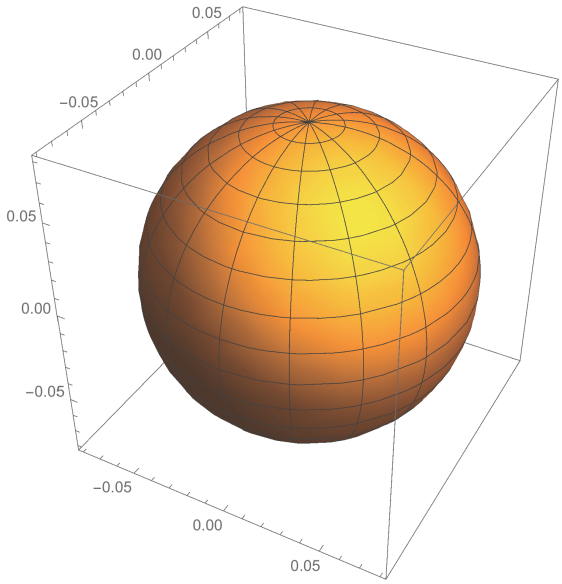
A fiber density distribution function is needed in the specification of a continuous fiber distribution.

4.3.3.1 Spherical

The fiber density distribution type "*spherical*" models an isotropic 3D distribution. This distribution corresponds to

$$R(\mathbf{n}) = \frac{1}{4\pi}.$$

It requires no additional parameters.



Example:

```
<distribution type="spherical">
</distribution>
```

4.3.3.2 Ellipsoidal

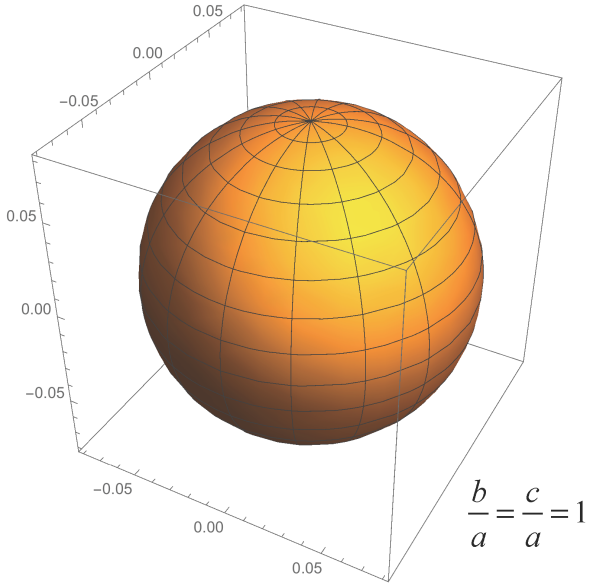
The fiber density distribution type "*ellipsoidal*" models a generally orthotropic 3D distribution. It corresponds to

$$R(\mathbf{n}) = C^{-1}\left[\left(\frac{n_1}{a}\right)^2 + \left(\frac{n_2}{b}\right)^2 + \left(\frac{n_3}{c}\right)^2\right]^{-1/2},$$

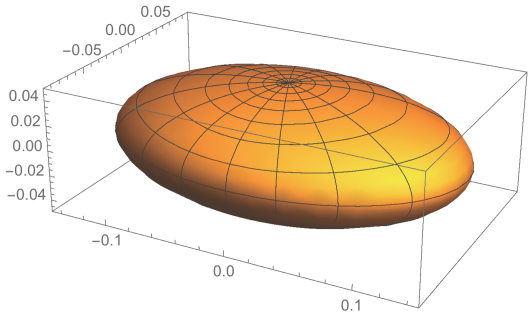
where (n_1, n_2, n_3) are the components of $\mathbf{n}$ in the local Cartesian basis $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ of each element, when specified (see Section 4.1.1) or the global basis by default; and C is calculated to satisfy the integration constraint on $R(\mathbf{n})$. The parameters (a, b, c) represents the semi-principal axes of the ellipsoid and must be positive. The following material parameters need to be defined:

<spa>	The semi-principal axes (a, b, c) of the ellipsoid	[ ]

The value of C is automatically adjusted to account for the values of the semi-principal axes (a, b, c). Therefore, only the relative ratios of these parameters matter.



$$\frac{b}{a} = \frac{c}{a} = 1$$



$$\frac{b}{a} = \frac{2}{3}, \frac{c}{a} = \frac{1}{3}$$

Example:

```
<distribution type="ellipsoidal">
  <spa>3,2,1</spa>
</distribution>
```

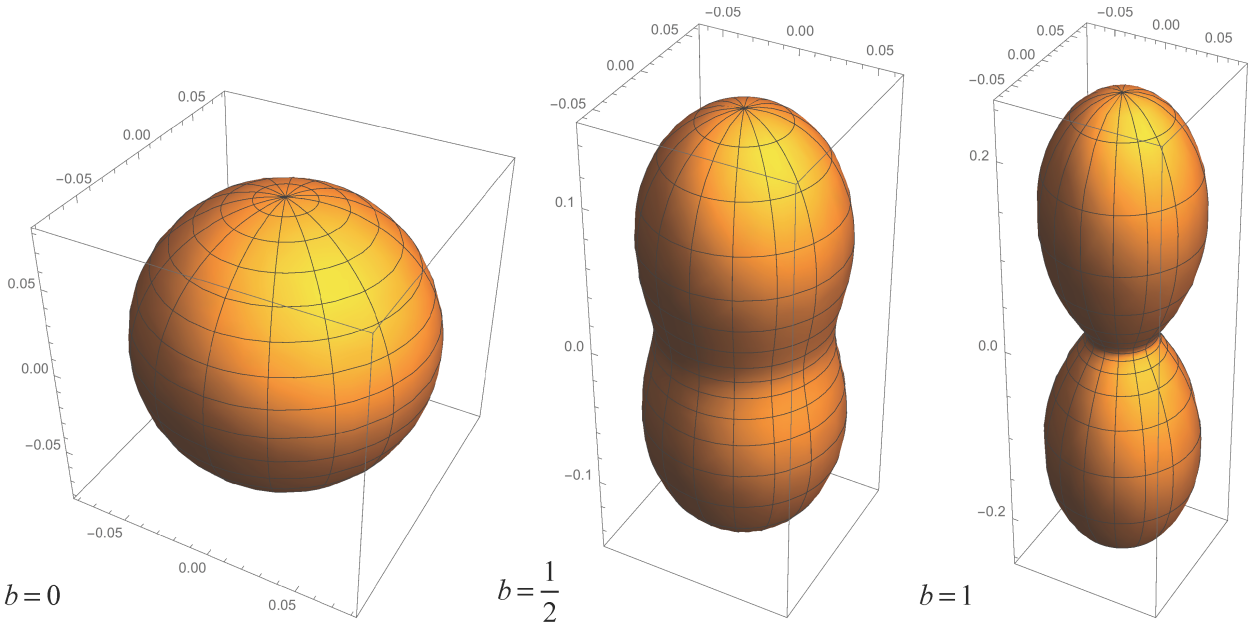
4.3.3.3 $\pi-$Periodic von Mises Distribution

The fiber density distribution type "*von-Mises-3d*" models a transversely isotropic 3D distribution [35]. It corresponds to

$$R(\mathbf{n}) = \frac{1}{\pi}\sqrt{\frac{b}{2\pi}}\frac{\exp\left(2bn_1^2\right)}{\text{erfi}\left(\sqrt{2b}\right)},$$

where (n_1, n_2, n_3) are the components of $\mathbf{n}$ in the local Cartesian basis $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ of each element, when specified (see Section 4.1.1) or the global basis by default; and b is the concentration parameter ($b > 0$). The following material parameters need to be defined:

<b>	The concentration parameter b	[ ]



Example:

```
<distribution type="von-Mises-3d">
  <b>0.5</b>
</distribution>
```

4.3.3.4 Circular

The fiber density distribution type "*circular*" models a transversely isotropic 2D distribution. This distribution corresponds to

$$R(\mathbf{n}) = \frac{1}{2\pi}.$$

It requires no additional parameters.

Example:

```
<distribution type="circular">
</distribution>
```

4.3.3.5 Elliptical

The fiber density distribution type "*elliptical*" models an orthotropic 2D distribution. This distribution corresponds to

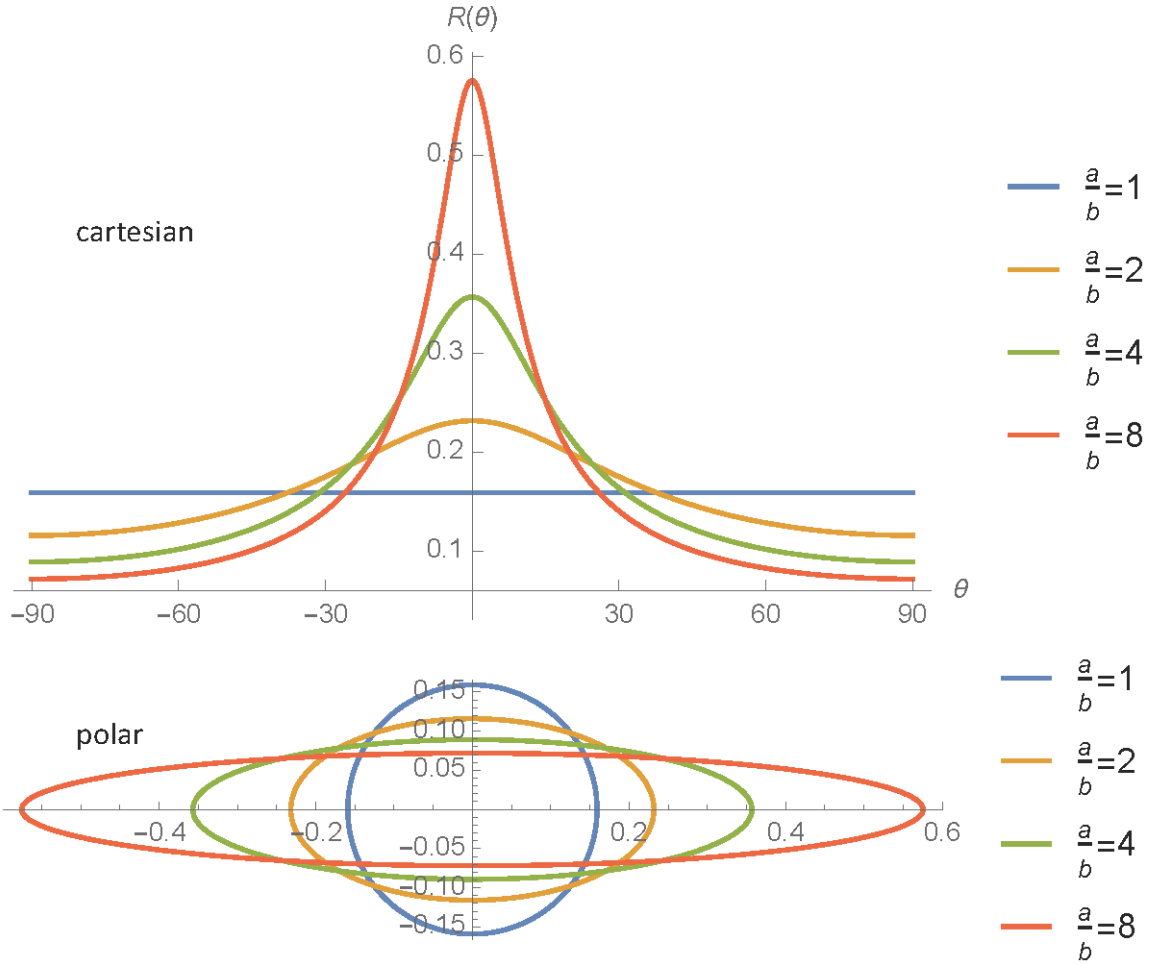
$$R(\mathbf{n}) = C^{-1}\left[\left(\frac{n_1}{a}\right)^2 + \left(\frac{n_2}{b}\right)^2\right]^{-1/2}$$

where (n_1, n_2, n_3) are the components of $\mathbf{n}$ in the local Cartesian basis $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ of each element, when specified (see Section 4.1.1) or the global basis by default, implying that the elliptical distribution lies in the $\{\mathbf{e}_1, \mathbf{e}_2\}$ plane; and (a, b) are the semi-principal axes of the ellipse. Here, $C = 4bK(e)$ where K is the complete elliptic integral of the first kind and

$$e = \sqrt{1 - \frac{b^2}{a^2}}$$

is the ellipse eccentricity. The following material parameters need to be defined:

<spa1>	The semi-principal axis a of the ellipse	[ ]
<spa2>	The semi-principal axis b of the ellipse	[ ]



Example:

```
<distribution type="elliptical">
  <spa1>8</spa1>
  <spa2>1</spa2>
</distribution>
```

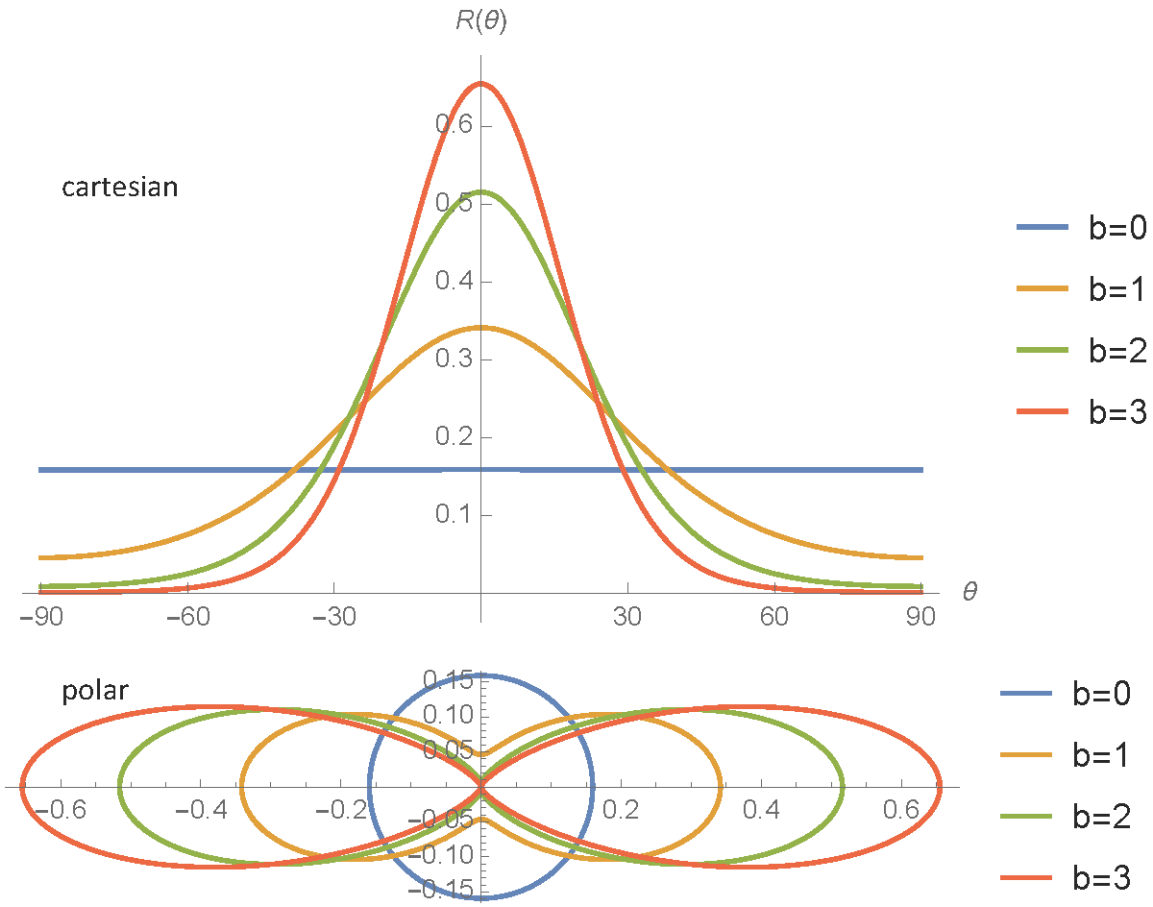
4.3.3.6 von Mises Distribution

The fiber density distribution type "*von-Mises-2d*" models an orthotropic 2D distribution. This distribution corresponds to

$$R(\mathbf{n}) = \frac{\exp\left[b\left(2n_1^2 - 1\right)\right]}{2\pi I_0(b)}$$

where (n_1, n_2, n_3) are the components of $\mathbf{n}$ in the local Cartesian basis $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ of each element, when specified (see Section 4.1.1) or the global basis by default, with the distribution lying in the $\{\mathbf{e}_1, \mathbf{e}_2\}$ plane; and b is the concentration parameter ($b > 0$). I_0 is the modified Bessel function of the first kind of order 0. The following material parameters need to be defined:

<b>	The concentration parameter b	[ ]



Example:

```
<distribution type="von-Mises-2d">
  <b>3</b>
</distribution>
```

4.3.4 Scheme

A numerical integration scheme is needed in the specification of a continuous fiber distribution to perform the integration over the unit sphere (3D) or the unit circle (2D). Use the uncoupled version of the scheme when modeling an uncoupled continuous fiber distribution.

4.3.4.1 Gauss-Kronrod Trapezoidal Rule

The scheme type for the Gauss-Kronrod trapezoidal rule is *"fibers-3d-gkt"* for unconstrained and uncoupled continuous fiber distributions. This integration rule should only be used with 3D fiber density distributions. This scheme automatically limits the range of integration to fibers that are in tension [43]. A Gauss-Kronrod quadrature rule is employed for integration across latitudes of the unit sphere. A trapezoidal rule is used for integration across longitudes. The following material parameters need to be defined:

<nph>	Number of integration points across latitudes	[ ]
<nth>	Number of integration points across longitudes	[ ]

The parameter <nph> must be one of the values 7, 11, 15, 19, 23 and 27. The parameter <nth> may be any number greater than 0. Odd values for <nth> produce more accurate results than even values. A recommended combination is nph=7 and nth=31. The total number of integration points is $N =$nph ×nth. Increasing values of N require increasing computational time.

Example:

```
<scheme type="fibers-3d-gkt">
  <nph>7</nph>
  <nth>31</nth>
</scheme>
```

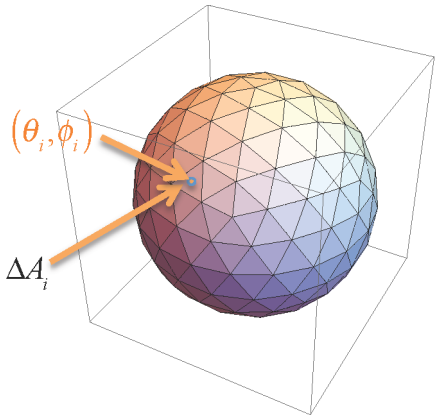
4.3.4.2 Finite Element Integration Rule

The scheme type for the finite element integration rule is *"fibers-3d-fei"* for unconstrained and uncoupled continuous fiber distributions. This integration rule should only be used with 3D fiber density distributions. This scheme discretizes the unit sphere into a set of N spherical triangles of nearly identical surface areas. The unit normal $\mathbf{n}$ passes through the centroid of each surface element. The integration is performed as a summation over N. For each direction $\mathbf{n}$ the stress is evaluated only if the fiber bundle is in tension along that direction. The following material parameters need to be defined:

<resolution>	the number of integration points N	[ ]

The parameter <resolution> must be one of the values 20, 34, 60, 74, 196, 210, 396, 410, 596, 610, 796, 810, 996, 1010, 1196, 1210, 1396, 1410, 1596, 1610, and 1796. A recommended combination is nph=7 and nth=31. A conservative choice for producing accurate results under general loading conditions is N =1610. Increasing values of N require increasing computational time.

$$I \approx \sum_{i=1}^{N} f\left(\theta_i, \phi_i\right) \Delta A_i$$



Example :

```
<scheme type="fibers-3d-fei">
  <resolution>1610</resolution>
</scheme>
```

4.3.4.3 Trapezoidal Rule

The scheme type for the trapezoidal integration rule is *"fibers-2d-trapezoidal"* for unconstrained and uncoupled continuous fiber distributions. This integration rule should only be used with 2D fiber density distributions. This scheme discretizes the unit circle into a set of N circular arcs of identical lengths. The unit normal $\mathbf{n}$ passes through the centroid of each arc element. The integration is performed as a summation over N. For each direction $\mathbf{n}$ the stress is evaluated only if the fiber bundle is in tension along that direction. The following material parameters need to be defined:

<nth>	Number of integration points N	[ ]

The parameter <nth> may be any number greater than 0. Odd values for <nth> produce more accurate results than even values.

Example:

```
<scheme type="fibers-2d-trapezoidal">
  <nth>31</nth>
</scheme>
```

4.4 Viscoelastic Solids

4.4.1 Uncoupled Viscoelastic Materials

These materials produce a viscoelastic response only for the deviatoric stress response. They must be used whenever the elastic response is uncoupled, as in the materials described in Section 4.1.2.

The material type for these materials is “uncoupled viscoelastic”. The following parameters need to be defined:

parameter	description	default value	units
<g0>	viscoelastic coefficient γ_0	1	[ ]
<t1>-<t6>	relaxation times	1	[**t**]
<g1>-<g6>	viscoelastic coefficients	0	[ ]
<elastic>	elastic component (must be an uncoupled elastic solid)		

$$\mathbf{S}\left(t\right)=\int_{-\infty}^{t}G\left(t-s\right)\frac{d\mathbf{S}^{e}}{ds}\,ds$$

For a uncoupled viscoelastic material, the second Piola Kirchhoff stress can be written as follows [65]:

$$\mathbf{S}\left(t\right)=pJ\,\mathbf{C}^{-1}+J^{-2/3}\int_{-\infty}^{t}G\left(t-s\right)\frac{d\left(\mathrm{Dev}\left[\tilde{\mathbf{S}}^{e}\right]\right)}{ds}\,ds\,,$$

where $\tilde{\mathbf{S}}^{e}$ is the elastic stress derived from $\tilde{W}\left(\tilde{\mathbf{C}}\right)$ (see Section 4.1.2) and G is the relaxation function. It is assumed that the relaxation function is given by the following discrete relaxation spectrum:

$$G\left(t\right)=\gamma_0+\sum_{i=1}^{N}\gamma_i\exp\left(-t/\tau_i\right),$$

Note that the user does not have to enter all the τ_i and γ_i coefficients. Instead, only the values that are used need to be entered. So, if N is 2, only τ_1, τ_2, γ_1 and $\gamma_2$have to be entered.

The *elastic* parameter describes the elastic material as given in Section 4.1.2.

Example:

```
<material id="1" name="Material 1" type="uncoupled viscoelastic">
  <g1>0.95</g1>
  <t1>0.01</t1>
  <k>100</k>
  <elastic type="Mooney-Rivlin">
    <c1>1</c1>
    <c2>0.0</c2>
  </elastic>
</material>
```

4.4.2 Unconstrained Viscoelastic Materials

The material type for viscoelastic materials is "*viscoelastic*". The following parameters need to be defined:

parameter	description	default value	units
<g0>	viscoelastic coefficient γ_0	1	[ ]
<t1>-<t6>	relaxation times	1	[**t**]
<g1>-<g6>	viscoelastic coefficients	0	[ ]
<elastic>	elastic component (must be an unconstrained elastic solid)		

For a viscoelastic material, the second Piola Kirchhoff stress can be written as follows [65]:

$$\mathbf{S}(t) = \int_{-\infty}^{t} G(t-s) \frac{d\mathbf{S}^e}{ds} ds ,$$

where $\mathbf{S}^e$ is the elastic stress and G is the relaxation function. It is assumed that the relaxation function is given by the following discrete relaxation spectrum:

$$G(t) = \gamma_0 + \sum_{i=1}^{N} \gamma_i \exp(-t/\tau_i) ,$$

Note that the user does not have to enter all the τ_i and γ_i coefficients. Instead, only the values that are used need to be entered. So, if N is 2, only τ_1, τ_2, γ_1 and $\gamma_2$have to be entered.

The *elastic* parameter describes the elastic material as given in Section 4.1.4.

Example:

```
<material id="1" name="Material 1" type="viscoelastic">
  <g1>0.95</g1>
  <t1>0.01</t1>
  <elastic type="neo-Hookean">
    <E>1</E>
    <v>0.0</v>
  </elastic>
</material>
```

4.5 Reactive Viscoelastic Solid

Reactive viscoelasticity models a material as a mixture of strong bonds, which are permanent, and weak bonds, which break and reform in response to loading [12]. Strong bonds produce the equilibrium elastic response, whereas weak bonds produce the transient viscous response. For a compressive reactive viscoelastic solid, the strain energy density is given by

$$\Psi_r\left(\mathbf{F}\right) = \Psi_r^e\left(\mathbf{F}\right) + \sum_u w^u \Psi_0^b\left(\mathbf{F}^u\right)$$

where Ψ_r^e is the strain energy density of strong bonds and Ψ_0^b is the strain energy density of weak bonds, when they all belong to the same generation. $\mathbf{F}$ is the deformation gradient of the strong bonds and the initial weak bond generation, wherease $\mathbf{F}^u$ is the relative deformation gradient for the $u-$generation weak bonds, such that $\mathbf{F}^u = \mathbf{R}\left(u\right)$ (the rotation tensor in the polar decomposition of $\mathbf{F}$) at time u. In this expression, $w^u\left(\mathbf{X},t\right)$ is the mass fraction of $u-$generation weak bonds, which evolves over time as described next. The summation is taken over all generations u that were created prior to the current time t.

Any number of valid solutions may exist for w^u, based on constitutive assumptions for the weak bond mass fraction supply $\hat{w}^u$. In particular, for $u-$generation bonds reforming in an unloaded state during the time interval $u \leqslant t < v$, and subsequently breaking in response to loading at $t = v$, Type I bond kinetics provides a solution of the form

$$w^u\left(\mathbf{X},t\right) = \begin{cases} 0 & t < u \\ f^u\left(\mathbf{X},t\right) & u \leqslant t < v \\ f^u\left(\mathbf{X},v\right) g\left(\mathbf{F}\left(v\right);\mathbf{X},t-v\right) & v \leqslant t \end{cases}$$

where

$$f^u\left(\mathbf{X},t\right) = 1 - \sum_{\gamma<u} w^\gamma\left(\mathbf{X},t\right)$$

and $g\left(\mathbf{F}\left(v\right);\mathbf{X},t-v\right)$ is a reduced relaxation function which may assume any number of valid forms. (A reduced relaxation function $g\left(t\right)$ satisfies $g\left(0\right) = 1$ and $g\left(t\to\infty\right) = 0$, and decreases monotonically with t.) In particular, g may depend on the strain at time v relative to the reference configuration of the $u-$generation. In the recursive expression above, the earliest generation $u = -\infty$, which is initially at rest, produces $w^u\left(t\right) = 1$ for $t < v$ and $w^u\left(t\right) = g\left(\mathbf{F}^u\left(v\right);\mathbf{X},t-v\right)$ for $t \geqslant v$; this latter expression seeds the recursion for subsequent generations. Therefore, providing a functional form for g suffices to produce the solution for all bond generations u.

For Type II bond kinetics, the solution for the mass fractions is given by

$$w^u\left(t\right) = \begin{cases} 0 & t < u \\ 1 - g\left(t-u\right) & u \leqslant t < v \\ g\left(t-v\right) - g\left(t-u\right) & v \leqslant t \end{cases} .$$

For this type of bond kinetics, the reduced relaxation function g cannot depend on the magnitude of the strain, because strain-dependence might violate the constraint $0 \leqslant w^u \leqslant 1$. Thus, type II bond kinetics is only valid for quasilinear viscoelasticity, whereas type I bond kinetics also encompasses nonlinear viscoelasticity.

For all bond kinetics, it is also possible to constrain the occurrence of the breaking-and-reforming reaction to specific forms of the strain. For example, the reaction may be allowed to proceed only in the case of dilatational strain, or only in the case of distortional strain.

For a material with an uncoupled formulation, the strain energy density has the form

$$\Psi_r(\mathbf{F}) = U(J) + \tilde{\Psi}_r^e\left(\tilde{\mathbf{F}}\right) + \sum_u w^u \tilde{\Psi}_0^b\left(\tilde{\mathbf{F}}^u\right)$$

where $\tilde{\mathbf{F}} = J^{-1/3}\mathbf{F}$.

The material type for a compressive reactive viscoelastic solid is "reactive viscoelastic". For the uncoupled formulation the material type is "uncoupled reactive viscoelastic". The following parameters need to be defined:

<kinetics>	Bond kinetics type
<trigger>	Strain invariants that trigger weak bond breakage and reformation
<wmin>	Bond mass fraction threshold $w_{\min}$ for triggering new generation or merging and culling oldest
<emin>	Strain threshold $e_{\min}$ for triggering new generation ($e_{\min} \geq 0$)
<elastic>	Elastic (strong bond) material
<bond>	Weak bond material
<relaxation>	Reduced relaxation function
<recruitment>	Weak bond recruitment function

The <kinetics> parameter should be set to 1 for Type I bond kinetics or 2 for Type II bond kinetics. The <trigger> parameter should be set 0 when weak bonds break and reform in response to any form of the strain; it should be set to 1 when the trigger is distortional strain; and it should be set to 2 when the trigger is dilatational strain. The <elastic> and <bond> materials may be selected from the list of unconstrained elastic materials given in Section 4.1.4 (for "reactive viscoelastic") or from the list of uncoupled elastic materials in Section 4.1.2 (for "uncoupled reactive viscoelastic"). The <relaxation> material may be selected from the list provided in Section 4.5.1. The <recruitment> material may be selected from the list provided in Section 4.6.2.

The parameters <wmin> and <emin> can be adjusted to improve computational efficiency by reducing the number of generations u in an analysis. In practice, the value of <emin> should be no greater than one-hundredth of the equilibrium strain in a loading configuration. The parameter <wmin> must be in the range $0 \leq w_{\min} \leq 1$. In practice, a good balance between accuracy and efficiency is achieved by using $w_{\min} \leq 0.05$.

Example:

```
<material id="1" name="RV solid" type="reactive viscoelastic">
  <kinetics>1</kinetics>
  <trigger>0</trigger>
  <wmin>0.05</wmin>
  <emin>0.001</emin>
  <elastic type="Holmes-Mow">
    <density>1</density>
    <E>1</E>
    <v>0.3</v>
    <beta>0.5</beta>
  </elastic>
  <bond type="Holmes-Mow">
    <density>1</density>
```

```
    <E>1</E>
    <v>0.3</v>
    <beta>0.5</beta>
  </bond>
  <relaxation type="relaxation-exponential">
    <tau>4</tau>
  </relaxation>
  <recruitment type="CDF power">
    <mu0>1</mu0>
    <mu1>30</mu1>
    <alpha>2</alpha>
    <scale>1</scale>
</material>
```

4.5.1 Relaxation Functions

4.5.1.1 Exponential

The material type for this relaxation function is “relaxation-exponential”. The following material parameters need to be defined:

<tau>	Characteristic relaxation time τ	**[t]**

The reduced relaxation function for this material type is given by

$$g(t) = e^{-t/\tau}$$

4.5.1.2 Exponential Distortional

The material type for this relaxation function is “relaxation-exp-distortion”. The following material parameters need to be defined:

<tau0>	Characteristic relaxation time τ_0	**[t]**
<tau1>	Characteristic time τ_1	
<alpha>	Power exponent α	

The reduced relaxation function for this material type is given by

$$g(\mathbf{F}(v); t - v) = e^{-(t-v)/\tau[K_2(v)]}$$

where

$$\tau(K_2(v)) = \tau_0 + \tau_1 \cdot (K_2(v))^{\alpha}$$

In general, $K_2 = |\mathrm{dev}\,\boldsymbol{\eta}|$ where $\boldsymbol{\eta} = \ln \mathbf{V}$ is the spatial natural (left Hencky) strain tensor and $\mathbf{V}$ is the left stretch tensor. K_2 is evaluated at the time v when weak bonds from the $u-$generation start breaking.

4.5.1.3 Exponential Distortional User-Specified

The material type for this relaxation function is “relaxation-exp-dist-user”. The following material parameters need to be defined:

<tau type="math | point">	Characteristic relaxation time τ_0	[**t**]

The reduced relaxation function for this material type is given by

$$g\left(\mathbf{F}\left(v\right);t-v\right)=e^{-(t-v)/\tau[K_2(v)]}$$

where $\tau\left(K_2\left(v\right)\right)$ is given as a mathematical expression (type="math"), or as a loadcurve (type="point").

Example:

```
<material id="1" name="RVE" type="reactive viscoelastic">
  <kinetics>1</kinetics>
  <trigger>1</trigger>
  <elastic type="neo-Hookean">
    <density>1</density>
    <E>0.13</E>
    <v>0.3</v>
  </elastic>
  <bond type="neo-Hookean">
    <density>1</density>
    <E>0.52</E>
    <v>0.3</v>
  </bond>
  <relaxation type="relaxation-exp-dist-user">
    <tau type="math">1.0+2.0*(K2^0.5)</tau>
  </relaxation>
</material>
```

Example:

```
<material id="1" name="RVE UC" type="uncoupled reactive viscoelastic">
  <kinetics>1</kinetics>
  <trigger>1</trigger>
  <k>25</k>
  <elastic type="Mooney-Rivlin">
    <density>1</density>
    <c1>0.025</c1>
    <c2>0</c2>
  </elastic>
  <bond type="Mooney-Rivlin">
    <density>1</density>
    <c1>0.025</c1>
    <c2>0</c2>
  </bond>
  <relaxation type="relaxation-exp-dist-user">
```

```
      <tau type="point">
        <interpolate>SMOOTH</interpolate>
        <points>
          <point>0.00, 1.00</point>
          <point>0.25, 2.00</point>
          <point>0.50, 2.41</point>
          <point>0.75, 2.73</point>
          <point>1.00, 3.00</point>
        </points>
      </tau>
    </relaxation>
  </material>
```

4.5.1.4 Exponential Continuous Spectrum

This relaxation function is derived from a continuous exponential relaxation spectrum, see *FEBio Theory Manual*. The material type for this relaxation function is "relaxation-CSexp". The following material parameter needs to be defined:

<tau>	Characteristic relaxation time τ	**[t]**

This parameter must satisfy $\tau > 0$. The reduced relaxation function for this material type is given by

$$g(t) = 2\sqrt{\frac{t}{\tau}}K_1\left(2\sqrt{\frac{t}{\tau}}\right)$$

where $K_1(z)$ is the modified Bessel function of the second kind, of order 1.

4.5.1.5 Exponential Continuous Spectrum Distortional User-Specified

See Section 4.5.1.4 for the description of this relaxation function. When the material parameters vary with the distortional strain K_2, the material type is "relaxation-CSexp-dist-user". The following material parameter needs to be defined:

<tau type="math | point">	Characteristic relaxation time τ	**[t]**

The definition of K_2 is given in Section 4.5.1.3. See Section 4.5.1.3 for examples of how to specify these functions.

4.5.1.6 Fung

This relaxation function, which is derived from a continuous relaxation spectrum, was introduced by Fung [31]. The material type for this relaxation function is "relaxation-Fung". The following material parameters need to be defined:

<tau1>	Characteristic relaxation time τ_1	**[t]**
<tau2>	Characteristic relaxation time τ_2	**[t]**

These parameters must satisfy $\tau_2 > \tau_1$. The reduced relaxation function for this material type is given by

$$g(t) = \frac{\tau_2 e^{-t/\tau_2} - \tau_1 e^{-t/\tau_1} + t\left[\mathrm{Ei}\left(-\frac{t}{\tau_2}\right) - \mathrm{Ei}\left(-\frac{t}{\tau_1}\right)\right]}{\tau_2 - \tau_1}$$

where $\mathrm{Ei}(\cdot)$ is the exponential integral function.

4.5.1.7 Malkin

This relaxation function is derived from the continuous relaxation spectrum given in [58]. The material type for this relaxation function is "relaxation-Malkin". The following material parameters need to be defined:

<tau1>	Characteristic relaxation time τ_1	**[t]**
<tau2>	Characteristic relaxation time τ_2	**[t]**
<beta>	Power exponent β	[ ]

These parameters must satisfy $\tau_2 > \tau_1 > 0$. When $\beta \neq 1$, the reduced relaxation function for this material type is given by

$$g(t) = \frac{(\beta - 1)\, t^{1-\beta}}{\tau_1^{1-\beta} - \tau_2^{1-\beta}} \left[\Gamma\left(\beta - 1, \frac{t}{\tau_2}\right) - \Gamma\left(\beta - 1, \frac{t}{\tau_1}\right)\right]$$

where $\Gamma(a, z)$ is the incomplete gamma function. In the limit as $\beta \to 1$ this function reduces to the form given by [34],

$$g(t) = \frac{\Gamma\left(0, \frac{t}{\tau_2}\right) - \Gamma\left(0, \frac{t}{\tau_1}\right)}{\ln \frac{\tau_2}{\tau_1}} = \frac{\mathrm{Ei}\left(-\frac{t}{\tau_2}\right) - \mathrm{Ei}\left(-\frac{t}{\tau_1}\right)}{\ln \frac{\tau_1}{\tau_2}}$$

where $\mathrm{Ei}(z)$ is the exponential integral function.

4.5.1.8 Malkin Distortional

See Section 4.5.1.7 for the description of this relaxation function. When the material parameters vary with the distortional strain K_2 according to

$$\tau_1(K_2) = \tau_{10} + \tau_{11} \exp\left(-\frac{K_2}{s_1}\right)$$
$$\tau_2(K_2) = \tau_{20} + \tau_{21} \exp\left(-\frac{K_2}{s_2}\right)$$

the material type is "relaxation-Malkin-distortion". The following material parameters need to be defined:

<t1c0>	Parameter τ_{10}	**[t]**
<t1c1>	Parameter τ_{11}	**[t]**
<t1s0>	Parameter s_1	[ ]
<t2c0>	Parameter τ_{20}	**[t]**
<t2c1>	Parameter τ_{21}	**[t]**
<t2s0>	Parameter s_2	[ ]
<beta>	Poiwer exponent β	[ ]

The definition of K_2 is given in Section 4.5.1.3. See Section 4.5.1.3 for examples of how to specify these functions.

4.5.1.9 Malkin Distortional User-Specified

See Section 4.5.1.7 for the description of this relaxation function. When the material parameters vary with the distortional strain K_2, the material type is "relaxation-Malkin-dist-user". The following material parameters need to be defined:

<tau1 type="math | point">	Characteristic relaxation time τ_1	**[t]**
<tau2 type="math | point">	Characteristic relaxation time τ_2	**[t]**
<beta type="math | point">	Power exponent β	[ ]

The definition of K_2 is given in Section 4.5.1.3. See Section 4.5.1.3 for examples of how to specify these functions.

4.5.1.10 Park

The material type for this relaxation function is "relaxation-Park". The following material parameters need to be defined:

<tau>	Characteristic relaxation time τ	**[t]**
<beta>	Power exponent β	[ ]

The reduced relaxation function for this material type is given by

$$g(t) = \frac{1}{1 + \left(\frac{t}{\tau}\right)^{\beta}}$$

4.5.1.11 Park Distortional

The material type for this relaxation function is "relaxation-Park-distortion". The following material parameters need to be defined:

<tau0>	Characteristic relaxation time τ_0	**[t]**
<beta0>	Power exponent at zero strain β_0	[ ]
<tau1>	Characteristic relaxation time τ_1	
<beta1>	Power exponent at zero strain β_1	
<alpha>	Power exponent α	

The reduced relaxation function for this material type is given by

$$g(\mathbf{F}(v); t - v) = \frac{1}{1 + \left(\frac{t-v}{\tau}\right)^{\beta}}$$

where

$$\tau = \tau_0 + \tau_1 \cdot (K_2(v))^{\alpha}$$

and

$$\beta = \beta_0 + \beta_1 \cdot (K_2(v))^{\alpha}$$

The definition of $K_2(v)$ is given in Section 4.5.1.3.

4.5.1.12 Park Distortional User-Specified

The material type for this relaxation function is “relaxation-Park-dist-user”. The following material parameters need to be defined:

<tau type="math | point">	Characteristic relaxation time τ	**[t]**
<beta type="math | point">	Power exponent β	[ ]

The reduced relaxation function for this material type is given by

$$g\left(\mathbf{F}\left(v\right);t-v\right)=\frac{1}{1+\left(\frac{t-v}{\tau}\right)^{\beta}}$$

where $\tau\left(K_2\left(v\right)\right)$ and $\beta\left(K_2\left(v\right)\right)$ are user-specified functions, given either as mathematical expressions or loadcurves. The definition of $K_2\left(v\right)$ is given in Section 4.5.1.3. See Section 4.5.1.3 for examples of how to specify these functions.

4.5.1.13 Power

The material type for this relaxation function is “relaxation-power”. The following material parameters need to be defined:

<tau>	Characteristic relaxation time τ	**[t]**
<beta>	Power exponent β	[ ]

The reduced relaxation function for this material type is given by

$$g\left(t\right)=\frac{1}{\left(1+\frac{t}{\tau}\right)^{\beta}}$$

4.5.1.14 Power Distortional

The material type for this relaxation function is “relaxation-power-distortion”. The following material parameters need to be defined:

<tau0>	Characteristic relaxation time τ_0	**[t]**
<beta0>	Power exponent at zero strain β_0	[ ]
<tau1>	Characteristic relaxation time τ_1	
<beta1>	Power exponent at zero strain β_1	
<alpha>	Power exponent α	

The reduced relaxation function for this material type is given by

$$g\left(\mathbf{F}\left(v\right);t-v\right)=\frac{1}{\left(1+\frac{t-v}{\tau}\right)^{\beta}}$$

where

$$\tau=\tau_0+\tau_1\cdot\left(K_2\left(v\right)\right)^{\alpha}$$

and

$$\beta=\beta_0+\beta_1\cdot\left(K_2\left(v\right)\right)^{\alpha}$$

The definition of $K_2\left(v\right)$ is given in Section 4.5.1.3.

4.5.1.15 Power

The material type for this relaxation function is “relaxation-power-dist-user”. The following material parameters need to be defined:

<tau type="math | point">	Characteristic relaxation time τ	[**t**]
<beta type="math | point">	Power exponent β	[ ]

The reduced relaxation function for this material type is given by

$$g\left(\mathbf{F}\left(v\right);t-v\right)=\frac{1}{\left(1+\frac{t-v}{\tau}\right)^{\beta}}$$

where $\tau\left(K_2\left(v\right)\right)$ and $\beta\left(K_2\left(v\right)\right)$ are user-specified functions, given either as mathematical expressions or loadcurves. The definition of $K_2\left(v\right)$ is given in Section 4.5.1.3. See Section 4.5.1.3 for examples of how to specify these functions.

4.5.1.16 Prony

The material type for this relaxation function is “relaxation-Prony”. The following material parameters need to be defined:

<g1>	Weight factor γ_1	[ ]
⋮	⋮	⋮
<g6>	Weight factor γ_6	[ ]
<t1>	Characteristic relaxation time τ_1	[**t**]
⋮	⋮	⋮
<t6>	Characteristic relaxation time τ_6	[**t**]

The reduced relaxation function for this material type is given by

$$g\left(t\right)=\frac{\sum_{i=1}^{6}\gamma_i e^{-t/\tau_i}}{\sum_{i=1}^{6}\gamma_i}$$

The coefficients γ_i are normalized by $\sum_i \gamma_i$ to enforce $g\left(0\right)=1$.

4.6 Reactive Damage Mechanics

A material may accumulate damage over a single cycle or multiple cycles of loading which alters its properties. In the classical framework of damage mechanics this attenuation of the material properties is described by a single scalar damage variable D when the material is isotropic ($0 \leqslant D \leqslant 1$). For anisotropic materials however, classical frameworks require that we introduce a function of the fourth-order damage tensor $\mathcal{D}$ to account for anisotropic damage. In FEBio, we use a reactive damage mechanics framework where the elastic response is proportional to the total number of intact bonds in the material and where, at any given time in the loading history, D represents the mass fraction of bonds that have broken. In this reactive framework, it is possible to also model damage in anisotropic materials by assuming that multiple bond types exist in the material, each of which may get damaged under different circumstances. Each bond type b may be described by a distinct solid constituent within a solid mixture (see Sections 4.1.2.16 and 4.1.4.26), each having its own scalar damage variable D^b.

For a given bond type, the strain energy density $\Psi_r(D, \mathbf{F})$ of a damaged material is given by

$$\Psi_r = (1 - D)\,\Psi_0\,,$$

where $\Psi_0(\mathbf{F})$ is the strain energy density when all bonds of that type are intact. Here, $1 - D$ represents the mass fraction of bonds that remains intact. Similarly, the Cauchy stress $\boldsymbol{\sigma}(D, \mathbf{F})$ of the damaged material is given by

$$\boldsymbol{\sigma} = (1 - D)\,\boldsymbol{\sigma}_0\,,$$

where $\boldsymbol{\sigma}_0(\mathbf{F})$ is the stress in the intact material, at a given strain, as derived from $\Psi_0(\mathbf{F})$. The intact material may be based on any of the elastic materials described in Sections 4.1.2 and 4.1.4.

The evolution of the damage variable D is determined by a user-selected scalar damage criterion measure $\Xi(\mathbf{F})$ (Ξ is the capital form of ξ). For example, $\Xi(\mathbf{F})$ may represent the strain energy density, or von Mises stress, or maximum principal normal strain, etc. If $\Xi(\mathbf{F})$ exceeds a given threshold at some state of deformation $\mathbf{F}$, then damage may initiate or progress further. If all bonds fail at a single threshold value Ξ_m, the material undergoes fracture. More commonly, bonds may fail with increasing probability as Ξ increases over a given range. Consequently, the evolution of damage may be based on a user-selected cumulative distribution function (c.d.f.) $F(\Xi)$, such that $D(t) = F(\Xi_m)$ where Ξ_m is the maximum value of Ξ achieved over the loading history up until the current time t.

4.6.1 General Specification of Damage Materials

The material types for damage materials are *"elastic damage"* and *"uncoupled elastic damage"*. The following parameters must be defined:

<elastic>	Specification of the elastic material
<damage>	Specification of the cumulative distribution function $F(\Xi)$
<criterion>	Specification of the damage criterion Ξ

The <elastic> tag encloses a description of the constitutive relation of the intact elastic material and associated material properties, as may be selected from the list provided in Section 4.1.4 for unconstrained materials (used with *"elastic damage"*) and Section 4.1.2 for uncoupled materials (used with *"uncoupled elastic damage"*). The <damage> tag encloses a description of the cumulative distribution function and associated material properties, as may be selected from the list presented in Section 4.6.2. The <criterion> tag encloses a description of the damage criterion, as may be selected from the list presented in Section 4.6.3.

Example:

```
<material id="1" type="elastic damage">
  <elastic type="neo-Hookean">
    <density>1</density>
    <E>0.13</E>
    <v>0.3</v>
  </elastic>
  <damage type="CDF Weibull">
    <alpha>8</alpha>
    <mu>0.3</mu>
    <Dmax>1</Dmax>
  </damage>
  <criterion type="DC max normal Lagrange strain">
  </criterion>
</material>
```

4.6.2 Cumulative Distribution Functions

Cumulative distribution functions provide the function $F(\Xi)$ that determines the evolution of the damage variable D based on the maximum value of the failure criterion Ξ up until the current time.

4.6.2.1 Simo

The material type for Simo's c.d.f. [71] is "*CDF Simo*". The following material parameters must be defined:

<a>	Parameter α (same units as Ξ, $\alpha > 0$)	[Ξ]
<b>	Parameter β ($0 \leqslant \beta \leqslant 1$)	[ ]

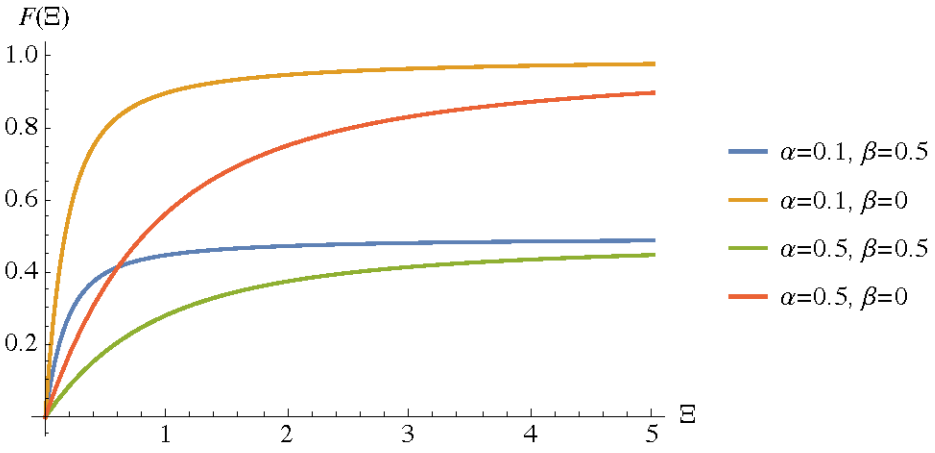
For this material the c.d.f. is given by

$$F(\Xi) = 1 - \beta - (1 - \beta)\frac{\alpha}{\Xi}\left(1 - e^{-\Xi/\alpha}\right) .$$

Note that

$$\lim_{\Xi \to \infty} F(\Xi) = 1 - \beta$$

represents the maximum allowable damage. Therefore, β regulates the maximum allowable damage, whereas α controls the rate at which $F(\Xi)$ increases with increasing Ξ.



Example:

```
<damage type="CDF Simo">
  <a>0.1</a>
  <b>0</b>
  <Dmax>1</Dmax>
</damage>
```

4.6.2.2 Log-Normal

The material type for a log-normal c.d.f. is "*CDF log-normal*". The following material parameters must be defined:

<mu>	Parameter μ (same units as Ξ, $\mu > 0$)	[Ξ]
<sigma>	Parameter σ ($\sigma > 0$)	[ ]
<Dmax>	Maximum allowable damage (optional, default is 1)	[ ]

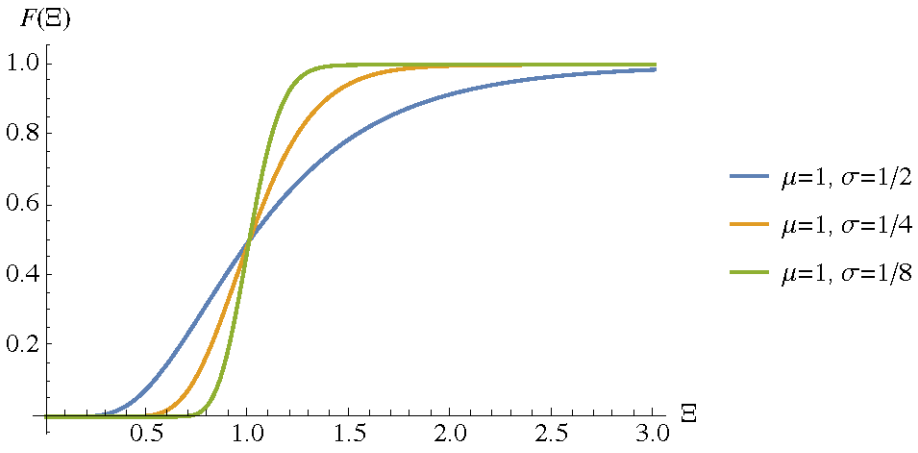
For this material the c.d.f. is given by

$$F(\Xi) = \frac{1}{2}\text{erfc}\left[-\frac{\ln(\Xi/\mu)}{\sigma\sqrt{2}}\right].$$

Note that

$$F(\mu) = 1/2,$$

which shows that μ is the value of Ξ at which half of the bonds break. Here, σ regulates the rate at which damage increases with increasing Ξ, with smaller values of σ producing a more rapid increase.



Example:

```
<damage type="CDF log-normal">
  <mu>1</mu>
  <sigma>0.5</sigma>
  <Dmax>1</Dmax>
</damage>
```

4.6.2.3 Weibull

The material type for a Weibull c.d.f. is "*CDF Weibull*". The following material parameters must be defined:

<mu>	Parameter μ (same units as Ξ, $\mu > 0$)	[Ξ]
<alpha>	Parameter α ($\alpha > 0$)	[ ]
<Dmax>	Maximum allowable damage (optional, default is 1)	[ ]

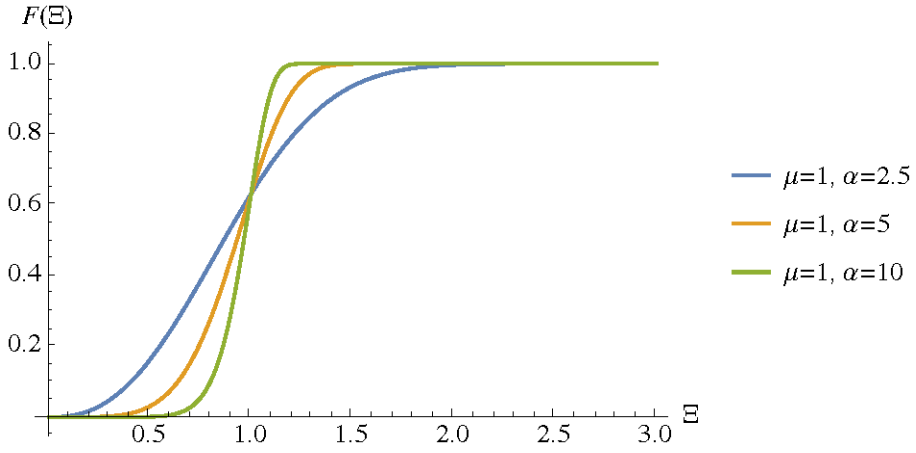
For this material the c.d.f. is given by

$$F(\Xi) = 1 - \exp\left[-(\Xi/\mu)^{\alpha}\right] .$$

Note that

$$F(\mu) = 1 - e^{-1} \approx 0.63 ,$$

which shows that μ is the value of Ξ at which the fraction $1 - e^{-1}$ of bonds break. Here, α regulates the rate at which damage increases with increasing Ξ, with higher values of α producing a more rapid increase.



Example:

```
<damage type="CDF Weibull">
  <mu>1</mu>
  <alpha>5.0</alpha>
  <Dmax>1</Dmax>
</damage>
```

4.6.2.4 Quintic Polynomial

The material type for a quintic polynomial c.d.f. is "*CDF quintic*". The following material parameters must be defined:

<mumin>	Parameter $\mu_{\min}$ (same units as Ξ, $\mu_{\min} > 0$)	[Ξ]
<mumax>	Parameter $\mu_{\max}$ (same units as Ξ, $\mu_{\max} > \mu_{\min}$)	[ ]
<Dmax>	Maximum allowable damage (optional, default is 1)	[ ]

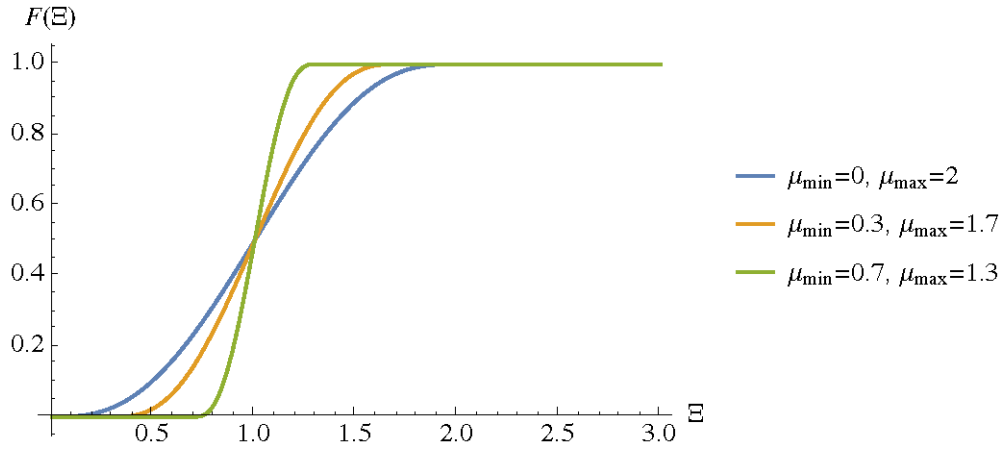
For this material the c.d.f. is given by

$$F(\Xi) = \begin{cases} 0 & \Xi \leqslant \mu_{\min} \\ x^3\left(10 - 15x + 6x^2\right) & \mu_{\min} < \Xi \leqslant \mu_{\max} \\ 1 & \mu_{\max} < \Xi \end{cases}, \quad x = \frac{\Xi - \mu_{\min}}{\mu_{\max} - \mu_{\min}}.$$

Note that

$$F\left(\frac{1}{2}\left(\mu_{\min} + \mu_{\max}\right)\right) = \frac{1}{2},$$

which shows that $\left(\mu_{\min} + \mu_{\max}\right)/2$ is the value of Ξ at which half of the bonds break. The range $\mu_{\max} - \mu_{\min}$ regulates the rate at which damage increases with increasing Ξ, with a narrower range producing a more rapid increase.



Example:

```
<damage type="CDF quintic">
  <mumin>0.3</mumin>
  <mumax>1.7</mumax>
  <Dmax>1</Dmax>
</damage>
```

4.6.2.5 Step

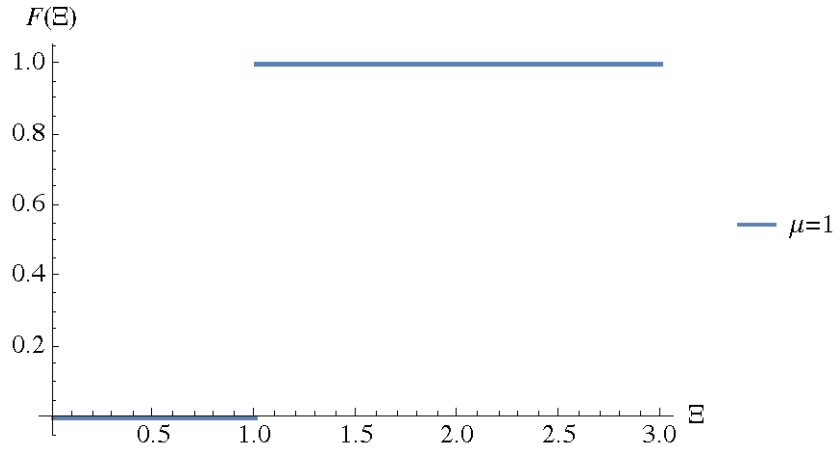
The material type for a step c.d.f. is "*CDF step*". The following material parameters must be defined:

<mu>	Parameter μ (same units as Ξ)	[Ξ]
<Dmax>	Maximum allowable damage (optional, default is 1)	[ ]

For this material the c.d.f. is given by

$$F(\Xi) = H(\Xi) \quad ,$$

where $H(\cdot)$ is the Heaviside unit step function. The step c.d.f. may be used to model fracture.



Example:

```
<damage type="CDF step">
  <mu>1.0</mu>
  <Dmax>1</Dmax>
</damage>
```

4.6.2.6 Power

The material type for a power c.d.f. is "*CDF power*". The following material parameters must be defined:

<mu0>	Parameter μ_0	[ ]
<mu1>	Parameter μ_1	[ ]
<alpha>	Power exponent α	[ ]
<scale>	Scale factor s	[Ξ]
<Dmax>	Maximum allowable damage (optional, default is 1)	[ ]

For this material the c.d.f. is given by

$$F(\Xi) = \mu_0 + \mu_1 \left(\frac{\Xi}{s}\right)^{\alpha}.$$

Example:

```
<damage type="CDF power">
  <mu0>0.1</mu0>
  <mu1>10</mu1>
  <alpha>0.5</alpha>
  <scale>100</scale>
  <Dmax>1</Dmax>
</damage>
```

4.6.3 Damage or Yield Criterion

The damage criterion provides the functional form of $\Xi(\mathbf{F})$ that determines the evolution of damage. It can also be used to define the yield criterion in the reactive plasticity framework. All the functions currently modeled in FEBio are defined over the range $\Xi \in [0, \infty[$.

4.6.3.1 Simo

The material type for Simo's damage criterion [71] is "*DC Simo*". For this criterion,

$$\Xi(\mathbf{F}) = \sqrt{2\Psi_0(\mathbf{F})}$$

Example:

```
<criterion type="DC Simo"/>
```

4.6.3.2 Strain Energy Density

The material type for strain energy density damage criterion is "*DC strain energy density*". For this criterion,

$$\Xi(\mathbf{F}) = \Psi_0(\mathbf{F})$$

Example:

```
<criterion type="DC strain energy density"/>
```

4.6.3.3 Specific Strain Energy

The material type for specific strain energy damage criterion is *"DC specific strain energy"*. For this criterion,

$$\Xi(\mathbf{F}) = \Psi_0(\mathbf{F})/\rho$$

where ρ is the elastic material's density.

Example:

```
<criterion type="DC specific strain energy"/>
```

4.6.3.4 Von Mises Stress

The material type for von Mises stress damage or yield criterion is *"DC von Mises stress"*. For this criterion,

$$\Xi(\mathbf{F}) = \sqrt{\frac{3}{2}\operatorname{dev}\boldsymbol{\sigma} : \operatorname{dev}\boldsymbol{\sigma}} = \sqrt{\frac{1}{2}\left[(\sigma_1-\sigma_2)^2 + (\sigma_2-\sigma_3)^2 + (\sigma_3-\sigma_1)^2\right]}$$

where $\sigma_1, \sigma_2, \sigma_3$ are the principal values of $\boldsymbol{\sigma}_0(\mathbf{F})$.

Example:

```
<criterion type="DC von Mises stress"/>
```

4.6.3.5 Maximum Shear Stress

The material type for maximum shear stress damage or yield criterion is *"DC max shear stress"*. For this criterion,

$$\Xi(\mathbf{F}) = \max\left(\frac{|\sigma_1 - \sigma_2|}{2}, \frac{|\sigma_2 - \sigma_3|}{2}, \frac{|\sigma_3 - \sigma_1|}{2}\right)$$

where $\sigma_1, \sigma_2, \sigma_3$ are the principal values of $\boldsymbol{\sigma}_0(\mathbf{F})$.

Example:

```
<criterion type="DC max shear stress"/>
```

4.6.3.6 Maximum Normal Stress

The material type for maximum normal stress damage or yield criterion is *"DC max normal stress"*. For this criterion,

$$\Xi(\mathbf{F}) = \max(\sigma_1, \sigma_2, \sigma_3)$$

where $\sigma_1, \sigma_2, \sigma_3$ are the principal values of $\boldsymbol{\sigma}_0(\mathbf{F})$.

Example:

```
<criterion type="DC max normal stress"/>
```

4.6.3.7 Drucker Shear Stress

The material type for the Drucker shear stress criterion is *"DC Drucker shear stress"*. It is based on the yield criterion for plasticity introduced in [29]. For this criterion,

$$\Xi(\mathbf{F}) = k = \left(J_2^3 - cJ_3^2\right)^{1/6}$$

where $J_2 = \frac{1}{2}\operatorname{dev}\boldsymbol{\sigma}_0 : \operatorname{dev}\boldsymbol{\sigma}_0$, $J_3 = \det(\operatorname{dev}\boldsymbol{\sigma}_0)$, k is the yield limit in simple shear and c is a user-specified non-dimensional material constant which must lie in the range $-\frac{27}{8} \leq c \leq \frac{9}{4}$. To better understand the meaning of k, consider uniaxial loading of a bar which yields at the normal stress value of σ_y. In this case,

$$k = \frac{\sigma_y}{\sqrt{3}}\left(1 - \frac{4}{27}c\right)^{1/6} \quad \frac{\sigma_y}{\sqrt{3}}\left(\frac{2}{3}\right)^{1/6} \leq k \leq \frac{\sigma_y}{\sqrt{3}}\left(\frac{3}{2}\right)^{1/6}$$

In the special case when $c = 0$ the Drucker criterion reduces to the von Mises criterion, with $k = \sigma_y/\sqrt{3}$.

Example:

```
<criterion type="DC Drucker shear stress">
  <c>2.25</c>
</criterion>
```

4.6.3.8 Maximum Normal Lagrange Strain

The material type for maximum normal Lagrange strain damage criterion is *"DC max normal Lagrange strain"*. For this criterion,

$$\Xi\left(\mathbf{F}\right) = \max\left(E_1, E_2, E_3\right)$$

where E_1, E_2, E_3 are the principal values of $\mathbf{E} = \left(\mathbf{F}^T \cdot \mathbf{F} - \mathbf{I}\right)/2$.

Example:

```
<criterion type="DC max normal Lagrange strain"/>
```

4.6.3.9 Octahedral Shear Strain

The material type for octahedral strain damage criterion is *"DC octahedral shear strain"*. For this criterion,

$$\Xi\left(\mathbf{F}\right) = \sqrt{\frac{2}{3}\operatorname{dev}\mathbf{E} : \operatorname{dev}\mathbf{E}}$$

where $\mathbf{E}$ is the Lagrange strain tensor.

4.7 Reactive Plasticity

4.7.1 Kinematic "Hardening" Response

Reactive plasticity models a material as a mixture of bonds that break in response to loading and reform in a stressed state [84]. This framework is based on constrained reactive mixtures of solids [7]. A reactively plastic solid consists of a mixture of n_f bond families, all of which share the same elastic response characterized by the user-defined strain energy density Ψ_0 (Section 4.1.4), each of which yields at a different threshold $\Phi_{m\beta}$ of a user-defined yield measure Φ (such as the von Mises stress, see Section 4.6.3). When all n_f bond families have yielded, the response becomes perfectly plastic, implying that the yield measure no longer increases with increasing strain. When $n^f > 1$ the stress-strain response exhibits the characteristic kinematic "hardening" response consistent with the Bauschinger effect of classical plasticity theory [75]. Optionally, an additional bond family may be included that never yields, remaining elastic over the entire loading history and imparting a linear "hardening" regime.

The deformation gradient of a reactive plasticity material is $\mathbf{F}^s$ and its associated mixture stress $\boldsymbol{\sigma}\left(\mathbf{F}^s\right)$ is evaluated as

$$\boldsymbol{\sigma}\left(\mathbf{F}^s\right) = \sum_\beta w_\beta \left(w_\beta^s \boldsymbol{\sigma}_0\left(\mathbf{F}^s\right) + w_\beta^y \boldsymbol{\sigma}_0\left(\mathbf{F}_\beta^y\right)\right) , \tag{4.7.1}$$

where w_β is the user-defined mass fraction of each bond family β in the mixture, w_β^s is the evolving mass fraction of intact bonds in family β which still exhibit an elastic response, and w_β^y is the evolving mass fraction of bonds in family β which have yielded (broken and reformed in a new reference configuration). These mass fractions satisfy $\sum_\beta w_\beta = 1$ and $w_\beta^s + w_\beta^y = 1$. The bond breaking-and-reforming reaction is denoted by $\mathcal{E}_\beta^s \rightarrow \mathcal{E}_\beta^y$, where $\mathcal{E}_\beta^s$ is the species associated with intact bonds while $\mathcal{E}_\beta^y$ is associated with yielded bonds, in bond family β. The mass fractions of bond family β satisfy $w_\beta^s = 1$ and $w_\beta^y = 0$ prior to yielding, and $w_\beta^s = 0$ and $w_\beta^y = 1$ after yielding. Here, $\mathbf{F}_\beta^y$ is the deformation gradient of yielded bonds β relative to the deformation when the yield threshold $\Phi_{m\beta}$ was first reached; it is calculated from the relation $\mathbf{F}^s = \mathbf{F}_\beta^y \cdot \mathbf{F}_\beta^{ys}$, where $\mathbf{F}_\beta^{ys}$ is a function of state that maps the yielded configuration $\mathbf{X}^y$ relative to the global reference configuration $\mathbf{X}^s$. The specific form of the constitutive model for $\mathbf{F}_\beta^{ys}$ used in FEBio can be found in [84] and the *FEBio Theory Manual*. Currently that constitutive model is not user-definable, though the user has the option to enforce isochoric plastic flow, implying that $\det \mathbf{F}_\beta^{ys} = 1$. The stress $\boldsymbol{\sigma}_0$ in the above relation for $\boldsymbol{\sigma}\left(\mathbf{F}^s\right)$ may be evaluated for any argument $\mathbf{F}$ as

$$\boldsymbol{\sigma}_0\left(\mathbf{F}\right) = \frac{1}{J}\frac{\partial \Psi_0}{\partial \mathbf{F}} \cdot \mathbf{F}^T , \tag{4.7.2}$$

where $J = \det \mathbf{F}$.

4.7.2 Plastic Flow Curve

Since each bond family behaves elastically until it yields, a family's yield threshold $\Phi_{m\beta}$ is generally not the value recorded on a stress-strain curve when the slope changes. That value may be called the *apparent yield threshold* Υ_β, which can be related to the true yield threshold $\Phi_{m\beta} = \Phi\left(\varepsilon_\beta\right)$, where ε_β is the true (logarithmic or Hencky) strain at which bond family β yields, by assuming a linear elastic stress-strain relationship prior to yielding, $\Phi\left(\varepsilon\right) = E\varepsilon$, where E is the elastic material's

Young's modulus. This assumption makes it possible to relate the bond family mass fractions w_β to the values of $\Phi_{m\beta}$ and Υ_β via

$$\Phi_{m\beta} = \Phi_{m,\beta-1} + \frac{\Upsilon_\beta - \Upsilon_{\beta-1}}{1 - \sum_{b=0}^{\beta-1} w_b}\,, \quad \Phi_{m0} = \Upsilon_0\,.$$

More details can be found in [84]. FEBio offers several different methods for specifying the plastic flow curve parameters Υ_β, which are used internally to evaluate $\Phi_{m\beta}$ and w_β:

4.7.2.1 User-Specified Flow Curve

A user-specified flow curve may be provided using the plastic flow curve material type "*PFC user*". The user enters a list of points representing $(\varepsilon_\beta, \Upsilon_\beta)$, where ε_β is the true strain corresponding to the apparent yield threshold Υ_β. Here, ε_β represents the total strain, not the plastic strain. The first point provided in this list, $(\varepsilon_0, \Upsilon_0)$, corresponds to the yield strain ε_0 and yield stress Υ_0 for this material, such that the ratio Υ_0/ε_0 is used internally to calculate Young's modulus E for the selected material used in this plasticity analysis. It is the user's responsibility to ensure this consistency. The number of bond families, n_f, is automatically set to the number of user-specified points on this flow curve. To maintain computational efficiency it is recommended to keep this number small, e.g., no greater than 30 points. Specifying only one point produces an elastic-perfectly plastic material response.

The response produced by the reactive plasticity material will agree nearly exactly with the user-specified points as long as the strain ε remains in the infinitesimal range. However, as the strain ε becomes finite, the reactive plasticity response may deviate non-negligibly from the user-specified flow curve, due to the nonlinearity of the hyperelastic relation describing the elastic response of the material. In practice, for isotropic elastic responses, using a "*natural neo-Hookean*" material (Section 4.1.4.18) will produce the least amount of deviation when using large strains, since its uniaxial stress-strain response is nearly linear with ε, $\Phi(\varepsilon) = E\varepsilon\, e^{-(1-2\nu)\varepsilon}$, where E is Young's modulus and ν is Poisson's ratio. As ν approaches $\frac{1}{2}$ this expression produces an exact linear response under uniaxial loading, $\Phi(\varepsilon) = E\varepsilon$, even under finite deformation. When using $\nu < \frac{1}{2}$, care must be taken not to exceed $\varepsilon_{\text{max}} = (1-2\nu)^{-1}$, since the slope of $\Phi(\varepsilon)$ becomes zero at that value and negative beyond it.

4.7.2.2 Mathematical Expression for Flow Curve

Alternatively, a mathematical expression may be supplied to describe $\Upsilon(\varepsilon)$, using the plastic flow curve material type "*PFC math*". The following parameters must also be defined:

<nf>	Number of bond families n_f ($n_f > 0$)	[ ]
<e0>	The value of ε_0 that produces $\Upsilon(\varepsilon_0) = \Upsilon_0$	[ ]
<emax>	The highest value of ε expected in this analysis	[ ]

It is the user's responsibility to ensure that the supplied mathematical function produces Υ_0/ε_0 equal to Young's modulus for the elastic material model being used. The actual response produced by the reactive plasticity material will agree nearly exactly with the mathematical expression for $\Upsilon(\varepsilon)$ as long as the strain ε remains in the infinitesimal range. However, as the strain ε becomes finite, the reactive plasticity response may deviate non-negligibly from the user-specified flow curve, due to the nonlinearity of the hyperelastic relation describing the elastic response of

the material. In practice, for isotropic elastic responses, using a "*natural neo-Hookean*" material (Section 4.1.4.18) will produce the least amount of deviation when using large strains, since its uniaxial stress-strain response is nearly linear with ε, $\Phi(\varepsilon) = E\varepsilon\, e^{-(1-2\nu)\varepsilon}$, where E is Young's modulus and ν is Poisson's ratio. As ν approaches $\frac{1}{2}$ this expression produces an exact linear response under uniaxial loading, $\Phi(\varepsilon) = E\varepsilon$, even under finite deformation. When using $\nu < \frac{1}{2}$, care must be taken not to exceed $\varepsilon_{\text{max}} = (1-2\nu)^{-1}$, since the slope of $\Phi(\varepsilon)$ becomes zero at that value and negative beyond it.

4.7.2.3 Custom Flow Curve

This custom method for specifying the flow curve was provided in the paper that introduced reactive plasticity [84], therefore it is called "*PFC paper*". The following parameters must be defined:

<nf>	Number of bond families n_f ($n_f > 0$)	[ ]
<Y0>	Lowest yield threshold Υ_0 (Figure 4.7.1a-b)	[Φ]
<Ymax>	Highest yield threshold Υ_{max} (Figure 4.7.1a-b)	[Φ]
<w0>	Fraction w_0 of bonds that yield at Υ_0 ($0 \leq w_0 \leq 1$)	[ ]
<we>	Fraction w_e of bonds than never yield ($0 \leq w_e \leq 1$)	[ ]
<r>	Bias factor r for consecutive increments of Υ_β over the range $[\Upsilon_0, \Upsilon_{\text{max}}]$	[ ]

This method assumes that Υ_β values are evenly distributed between the *initial yield threshold* Υ_0 (e.g., the yield stress) and a *final yield threshold* Υ_{max}, parameters which may be identified from a stress strain curve (Figure 4.7.1a-b). Beyond Υ_{max}, the material either behaves as if it is perfectly plastic (a scenario which may be valid around the ultimate strength, for example), or it transitions to a linear hardening regime.

The family mass fractions w_β govern the influence of each family on the overall material response. The simplest model for w_β involves specifying the mass fraction of the first yielding family w_0, which controls the slope of the initial post-yield response (Figure 4.7.1a), and then evenly weighting the rest of the bond families. In cases where the material transitions to a linear hardening regime, we can recover this behavior by adding one more bond family, $\beta = n_f$, that never yields, thus remaining elastic. The associated mass fraction w_β for $\beta = n_f$ is called the *elastic mass fraction* and denoted w_e; a non-zero value for this parameter may be specified whenever we wish to include linear hardening behavior (Figure 4.7.1b). The effect of the mass fraction parameters w_0 and w_e is explored parametrically in Figure 4.7.1c and Figure 4.7.1d, respectively. In general, most ductile materials have w_0 very close to unity, which provides hardening behavior over a finite strain range. As $w_0 \to 1$ the stress-strain behavior approaches perfect plasticity. In contrast, when $w_e = 0$, the material response becomes perfectly plastic once the final yield threshold Υ_{max} has been exceeded. As w_e increases, a region of linear hardening is seen on a plot of the true stress against strain. For most ductile materials, w_e is usually 0 or on the order of 0.001.

It is also possible to introduce a *bias factor* r, which allows a geometric progression rather than uniform spacing of the apparent yield thresholds and family mass fractions. The bias factor r has the effect of modifying the shape of the hardening region between Υ_0 and Υ_{max} (Figure 4.7.1b). The full set of material parameters is then given by $\{n_f, \Upsilon_0, \Upsilon_{\text{max}}, w_0, w_e, r\}$. Setting $r = 1$ recovers the uniform distribution. Figure 4.7.1a-b graphically describes the influence of each parameter on simplified stress-strain curves, showing how these parameters may be extracted from experimental data.

Default values in FEBio are $n_f = 1$, $w_0 = 1$,$w_e = 0$, $\Upsilon_0 = \Upsilon_{\text{max}} = 0$, $r = 0.9$.

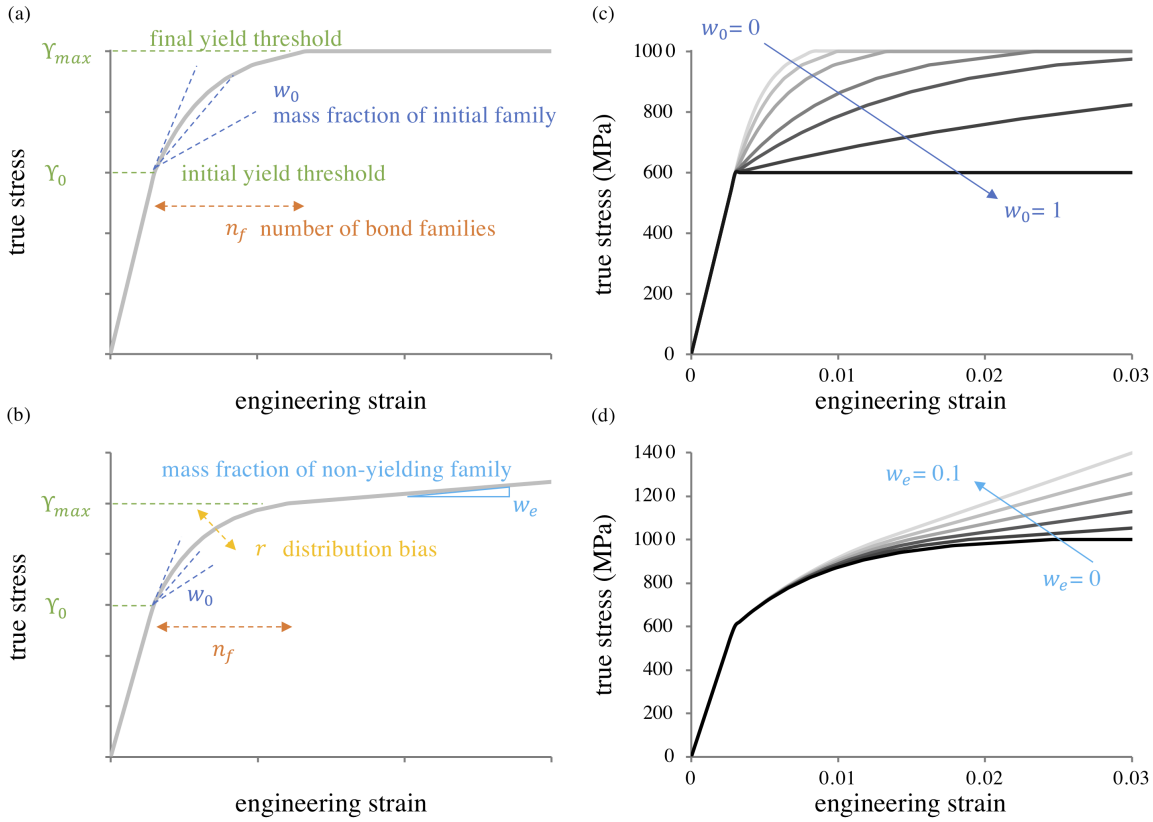


Figure 4.7.1: Modeling uniaxial stress-strain curves using the parameters given in Section 4.7.1. Identification of parameters on idealized stress-strain curves showing (a) a plateau in the stress, or (b) exhibiting a region of linear hardening. The yielding behavior is fully described by the set of parameters $\{n_f, \Upsilon_0, \Upsilon_{\text{max}}, w_0, w_e, r\}$. Parametric variations of (c) w_0 and (d) w_e illustrate their influence on the stress-strain response; other parameters are held fixed. In (c-d) $n_f = 10$, $\Upsilon_0 = 600$ MPa, $\Upsilon_{\text{max}} = 1000$ MPa, and $r = 1$. In (c), $w_e = 0$ and in (d) $w_0 = 0.75$.

4.7.3 Specification of Reactive Elasto-Plastic Solid

The material type of a reactive elasto-plastic solid with kinematic hardening is "*reactive plasticity*". The following parameters must be defined:

<isochoric>	Flag (0 or 1) for enforcing isochoric plastic flow (1 implies $\det \mathbf{F}_\beta^{ys} = 1, \forall\beta$) [ ]
<elastic>	Specification of the elastic material (Section 4.1.4)
<yield_criterion>	Specification of the yield criterion Φ (Section 4.6.3
<flow_curve>	Specification of the plastic flow curve (Section 4.7.2)

The <elastic> tag encloses a description of the constitutive relation of the intact elastic material and associated material properties, as may be selected from the list provided in Section 4.1.4 for unconstrained materials. The <yield_criterion> tag encloses a description of the yield criterion Φ, as may be selected from the list presented in Section 4.6.3. The <flow_curve> tag encloses a description of the relation between bond mass fractions w_β and apparent yield stresses Υ_β for the bond family β as described in Section 4.7.2.The default value in FEBio is isochoric=1.

Example: Idealized elastic-perfectly plastic response of steel (mm-N-s units)

```
<material id="1" type="reactive plasticity">
  <elastic type="neo-Hookean">
    <density>8.05e-9</density>
    <E>200e3</E>
    <v>0.30</v>
  </elastic>
  <yield_criterion type="DC von Mises stress"/>
  <flow_curve type="PFC paper">
    <Y0>450</Y0>
  </flow_curve>
</material>
```

Example: Annealed mild steel (mm-N-s units)

```
<material id="1" type="reactive plasticity">
  <isochoric>1</isochoric>
  <elastic type="neo-Hookean">
    <density>7.85e-9</density>
    <E>205e3</E>
    <v>0.29</v>
  </elastic>
  <yield_criterion type="DC von Mises stress"/>
  <flow_curve type="PFC paper">
    <nf>15</nf>
    <Y0>220</Y0>
    <Ymax>490</Ymax>
    <w0>0.973</w0>
    <we>0</we>
    <r>1</r>
  </flow_curve>
</material>
```

Example: Annealed copper (mm-N-s units)

```
<material id="1" type="reactive plasticity">
  <isochoric>1</isochoric>
  <elastic type="neo-Hookean">
    <density>7.764e-9</density>
    <E>120e3</E>
    <v>0.34</v>
  </elastic>
  <yield_criterion type="DC von Mises stress"/>
  <flow_curve type="PFC paper">
    <nf>15</nf>
    <Y0>60</Y0>
    <Ymax>288</Ymax>
    <w0>0.988</w0>
    <we>0</we>
    <r>1</r>
  </flow_curve>
</material>
```

Example: Unaged maraging steel (mm-N-s units)

```
<material id="1" type="reactive plasticity">
  <isochoric>1</isochoric>
  <elastic type="neo-Hookean">
    <density>8.00e-9</density>
    <E>165e3</E>
    <v>0.33</v>
  </elastic>
  <yield_criterion type="DC von Mises stress"/>
  <flow_curve type="PFC paper">
    <nf>22</nf>
    <Y0>398</Y0>
    <Ymax>1010</Ymax>
    <w0>0</w0>
    <we>0</we>
    <r>0.9</r>
  </flow_curve>
</material>
```

Example: Annealed aluminum 1100 (mm-N-s units)

```
<material id="1" type="reactive plasticity">
  <isochoric>1</isochoric>
  <elastic type="neo-Hookean">
    <density>2.71e-9</density>
    <E>68e3</E>
    <v>0.33</v>
  </elastic>
```

```
  <yield_criterion type="DC von Mises stress"/>
  <flow_curve type="PFC paper">
    <nf>18</nf>
    <Y0>63</Y0>
    <Ymax>112</Ymax>
    <w0>0.994</w0>
    <we>0</we>
    <r>0.6</r>
  </flow_curve>
</material>
```

Example: Mild steel (mm-N-s units)

```
<material id="1" type="reactive plasticity">
  <isochoric>1</isochoric>
  <elastic type="natural neo-Hookean">
    <density>2.71e-9</density>
    <E>206e3</E>
    <v>0.30</v>
  </elastic>
  <yield_criterion type="DC von Mises stress"/>
  <flow_curve type="PFC math">
    <nf>15</nf>
    <emin>0.0008403</emin>
    <emax>1.3</emax>
    <plastic_response>545.46*(0.011024+eps)^0.2589</plastic_response>
  </flow_curve>
</material>
```

Example: Steel (in-lbf-s units)

```
<material id="1" name="Steel" type="reactive plasticity">
  <density>1</density>
  <isochoric>1</isochoric>
  <elastic type="natural neo-Hookean">
    <E>29911000</E>
    <v>0.3</v>
  </elastic>
  <yield_criterion type="DC von Mises stress"/>
  <flow_curve type="PFC user">
  <plastic_response type="point">
    <interpolate>SMOOTH</interpolate>
    <points>
      <point> 0.002, 59822 </point>
      <point> 0.002841, 64450 </point>
      <point> 0.00469, 68500 </point>
      <point> 0.00953, 72000 </point>
      <point> 0.0193, 75000 </point>
    </points>
```

```
    </plastic_response>
  </flow_curve>
</material>
```

Example: elastic-perfectly plastic response, using PFC math flow curve

```
<material id="1" name="Elastoplastic Math" type="reactive plasticity">
  <density>1</density>
  <isochoric>1</isochoric>
  <elastic type="neo-Hookean">
    <E>200000</E>
    <v>0.3</v>
  </elastic>
  <yield_criterion type="DC von Mises stress"/>
  <flow_curve type="PFC math">
    <nf>1</nf>
    <e0>0.001</e0>
    <emax>1</emax>
    <plastic_response type="math">200</plastic_response>
  </flow_curve>
</material>
```

4.8 Reactive Elasto-Plastic Damage Mechanics

Plastic deformation (Section 4.7) is often coupled with damage, as the finite deformation and plastic flow of a loaded material typically induces some amount of failure. Within the constrained reactive mixture framework adopted in FEBio, damage is produced by bonds breaking permanently (Section 4.6), which reduces the generation mass fractions w^s_β or w^y_β [84]. In our treatment of elastoplastic damage we assume that both intact and yielded bonds may become damaged. Intact bonds belong to the $s-$generation which is present at $t = -\infty$. Once yielding occurs, all successive generations of that family are labeled as yielded bonds y. This distinction is necessary so we can then distinguish between damage to intact bonds (elastic damage) and damage to yielded bonds (plastic damage), since intact bonds which get damaged never have the ability to yield. Damage to intact bonds is represented by the reaction$\mathcal{E}^s_\beta \rightarrow \mathcal{E}^b_\beta$, where $\mathcal{E}^b_\beta$ is the species associated with broken bonds in bond family β. It may represent some initial damage value for a material with defects, or damage due to intermolecular failure of bonds that never yielded; this damage alters the value of w^s_β and we refer to it as *elastic damage*. Damage to yielded bonds represents *plastic damage*; it is represented by the reaction $\mathcal{E}^y_\beta \rightarrow \mathcal{E}^b_\beta$ and it alters the value of w^y_β. The mechanism of damage and the failure measure may be different for these two types of bonds, particularly since a stress- or energy-based failure measure may not be appropriate for plastic damage. It is important to note that the nature of the plastic deformation described in Section 4.7 remains unchanged. Damage modifies the material behavior by reducing the fraction of bonds in various generations, which scales the response accordingly.

In a reactive constrained mixture framework the insertion of damage into the reactive plasticity formulation is straightforward. Since bonds break permanently in a damage reaction, there is no need to define a function of state $\mathbf{F}^{ys}_\beta$ to describe a (non-existing) reformed configuration. Furthermore, the specific free energy of broken bonds is zero. The scalar *elastic damage criterion* $\Xi^e\left(\mathbf{F}^s\right)$, which is taken to have the same functional form for all bond families β, is the analog to the yield criterion Φ for plasticity. As shown in Section 4.6, the main contrast with reactive plasticity is that not all bonds in the family β break simultaneously at a single *elastic damage threshold* $\Xi^e_{m\beta}$. Instead, the fraction of broken bonds varies as a function of $\Xi^e\left(\mathbf{F}^s\right)$, denoted by $F^e\left(\Xi^e_\beta\right)$, such that $0 \leq F^e\left(\Xi^e_\beta\right) \leq 1$. Here, $F^e\left(\Xi^e_\beta\right)$ is a function of state; it must be a monotonically increasing function of its argument to satisfy the Clausius-Duhem inequality [61]. We may view F^e as a cumulative distribution function (CDF), whose corresponding probability distribution function (PDF) represents the probability of damage at a particular value of Ξ^e_β.

4.8.1 Theoretical Formulation

Here, we sketch the structure of the elastoplastic damage theory in FEBio. Since each bond family in reactive plasticity yields all at once, we can easily split an elastoplastic damage theory into two parts to represent elastic and plastic damage regimes. Assume that the first yielding reaction for bond family β occurs at time $t = u_\beta$. Prior to this initial yielding, the damage behavior described in Section 4.6 applies, and the material composition is generally a mixture of intact ($\mathcal{E}^s_\beta$) and broken ($\mathcal{E}^b_\beta$) bonds satisfying the reaction $\mathcal{E}^s_\beta \rightarrow \mathcal{E}^b_\beta$. The corresponding bond mass fractions satisfy $1 = w^s_\beta + w^b_\beta$ and $w^y_\beta = 0$, where $w^b_\beta = F^e\left(\Xi^e_\beta\right)$ is the elastic damage in bond family β. At $t = u_\beta$, the remaining intact bonds $w^s_\beta = 1 - w^b_\beta$ all yield, following the reaction $\mathcal{E}^s_\beta \rightarrow \mathcal{E}^y_\beta$. The family mass balance is then given as $1 = w^y_\beta + w^b_\beta$, since $w^s_\beta = 0$ after yielding.

For time $t > u_\beta$, yielded bonds may continue to yield, but they may also sustain damage

according to the reaction $\mathcal{E}^y_\beta \to \mathcal{E}^b_\beta$, which reduces their mass fraction w^y_β. Damage to yielded bonds may occur based on a function of state (often described as a plastic strain, though it is not an observable kinematic variable), which is distinct from the measure of elastic damage. Therefore, we denote the *plastic damage measure* as $\Xi^p\left(\mathbf{F}^{ys}_\beta\right)$ and its cumulative distribution function by $F^p\left(\Xi^p_\beta\right)$, under the assumption that all bond families β share the same functional forms for Ξ^p and F^p. For each bond family β, only the remaining undamaged fraction $1 - F^p\left(\Xi^p_\beta\right)$ of yielded bonds may break and reform as the next yielded generation.

It must be recognized that, just as $\mathbf{F}^{ys}_\beta$ is a constitutively-prescribed function of state and does not carry the meaning of a plastic deformation gradient, the plastic damage measure $\Xi^p\left(\mathbf{F}^{ys}_\beta\right)$ is also a function of state. This quantity is called a plastic strain for convenience only.

4.8.1.1 Stress and Damage

Recognizing that damaged (broken) bonds do not store free energy, the mixture stress is given by

$$\boldsymbol{\sigma} = \sum_\beta w_\beta \left(w^s_\beta \boldsymbol{\sigma}_0 \left(\mathbf{F}^s\right) + w^y_\beta \boldsymbol{\sigma}_0 \left(\mathbf{F}^y_\beta\right)\right) , \tag{4.8.1}$$

where the stresses $\boldsymbol{\sigma}_0$ are given by the standard hyperelasticity relation in Eq.(4.7.2). The reactive mixture equivalent of the damage variable D may be evaluated for elastoplastic damage as the fraction of all bonds that are broken,

$$D = w^b = \sum_\beta w_\beta w^b_\beta \, . \tag{4.8.2}$$

4.8.1.2 Damage Measures

For elastic damage, we may use the same functional measure as proposed for plastic yielding (e.g., the von Mises stress); this implies that the functions Ξ^e and Φ have the same form. For plastic damage of bonds in family β, we use the constitutively-determined mapping $\mathbf{F}^{ys}_\beta$ to define plastic right Cauchy-Green and Lagrange strain tensors through

$$\begin{aligned} \mathbf{C}^{ys}_\beta &= \left(\mathbf{F}^{ys}_\beta\right)^T \cdot \mathbf{F}^{ys}_\beta \\ \mathbf{E}^{ys}_\beta &= \frac{1}{2}\left(\mathbf{C}^{ys}_\beta - \mathbf{I}\right) \end{aligned} \, . \tag{4.8.3}$$

One possible constitutive relation for Ξ^p, which remains valid for general deformations, is to set it equal to the *effective plastic strain* e^p_β for the various bond families β,

$$e^p_\beta = \sqrt{\frac{2}{3} \operatorname{dev} \mathbf{E}^{ys}_\beta : \operatorname{dev} \mathbf{E}^{ys}_\beta} \, . \tag{4.8.4}$$

In a numerical implementation, the effective plastic strain e^p_0 of the first bond family to yield may be reported as the effective plastic strain in the entire material, for consistency with plastic strain measures in classical models of plasticity.

Quantities in this section do not represent plastic strains or plastic strain tensors, though we adopt the terminology due to similarities. Recall that the non-observable function of state $\mathbf{F}^{ys}_\beta =$

$\mathbf{F}_\beta^{ys}\left(\mathbf{F}^s, \rho_{r\beta}^\alpha\right)$ is a time-invariant mapping providing the reference configuration of a yielded bond y with respect to the reference configuration of the master constituent s, for family β. The quantities $\mathbf{C}_\beta^{ys}$ and $\mathbf{E}_\beta^{ys}$ then also represent non-observable functions of state calculated as strain tensors. Consequently, e_β^p is a measure of the relative motion of the reference configuration of bond family β , expressed as a scalar "strain". Physically, this amounts to the modeling assumption that once the breaking-and-reforming process takes a bond family out of a local neighborhood centered about its original position, the bond begins to degrade with further breaking-and-reforming processes. That each of these quantities exists for every bond family β emphasizes the lack of any true or unique plastic strain measure in this framework.

4.8.2 Specification of Reactive Elasto-Plastic Damage Solid

The material type of a reactive elasto-plastic damage solid with kinematic hardening is "*reactive plastic damage*". The following parameters must be defined:

<isochoric>	Flag (0 or 1) for enforcing isochoric plastic flow (1 implies $\det \mathbf{F}_\beta^{ys} = 1, \forall\beta$) [ ]
<elastic>	Specification of the elastic material (Section 4.1.4)
<yield_criterion>	Specification of the yield criterion Φ (Section 4.6.3
<flow_curve>	Specification of the plastic flow curve (Section 4.7.2)
<plastic_damage>	Specification of the plastic damage CDF $F^p\left(\Xi_\beta^p\right)$
<plastic_damage_criterion>	Specification of the plastic damage criterion $\Xi^p\left(\mathbf{F}_\beta^{ys}\right)$
<elastic_damage>	Specification of the elastic damage CDF $F^e\left(\Xi_\beta^e\right)$
<elastic_damage_criterion>	Specification of the elastic damage criterion $\Xi^e\left(\mathbf{F}^s\right)$

The <elastic> tag encloses a description of the constitutive relation of the intact elastic material and associated material properties, as may be selected from the list provided in Section 4.1.4 for unconstrained materials. The <yield_criterion> tag encloses a description of the yield criterion Φ, as may be selected from the list presented in Section 4.6.3. The <flow_curve> tag encloses a description of the relation between bond mass fractions w_β and apparent yield stresses Υ_β for the bond family β as described in Section 4.7.2. The plastic and elastic damage CDFs, in the <plastic_damage> and <elastic_damage> tags respectively, may be selected from Section 4.6.2, whereas the corresponding damage criteria <plastic_damage_criterion> and <elastic_damage_criterion> may be selected from Section 4.6.3. Default values are $n_f = 1$, $w_0 = 1$,$w_e = 0$, $\Upsilon_0 = \Upsilon_{\text{max}} = 0$, $r = 0.9$, and isochoric=1. If only plastic damage needs to be modeled, the tags <elastic_damage> and <elastic_damage_criterion> may be omitted. If <plastic_damage> and <plastic_damage_criterion> are also omitted the material response becomes identical to the "*reactive plasticity*" material described in Section 4.7.3.

Example: A-533 Grade B class 1 nuclear pressure vessel steel (mm-N-s units)

```
<material id="1" type="reactive plasticity">
  <isochoric>1</isochoric>
  <elastic type="neo-Hookean">
    <density>8.00e-9</density>
    <E>206.9e3</E>
    <v>0.29</v>
```

```
  </elastic>
  <yield_criterion type="DC von Mises stress"/>
  <flow_curve type="PFC paper">
    <nf>10</nf>
    <Y0>458</Y0>
    <Ymax>730</Ymax>
    <w0>0.985</w0>
    <we>0.00072</we>
    <r>1</r>
  </flow_curve>
  <plastic_damage type="CDF Weibull">
    <mu>3.3</mu>
    <alpha>3</alpha>
    <Dmax>1</Dmax>
  </plastic_damage>
  <plastic_damage_criterion type="DC octahedral shear strain"/>
</material>
```

4.9 Multigeneration Solids

This type of material [7] implements a mechanism for multigenerational interstitial growth of solids whereby each growth generation γ has a distinct reference configuration $\mathbf{X}^\gamma$ determined at the time t^γ of its deposition. Therefore, the solid matrix of a growing material consists of a multiplicity of intermingled porous bodies, each representing a generation γ, all of which are constrained to move together in the current configuration $\mathbf{x}$. The deformation gradient of each generation is $\mathbf{F}^\gamma = \partial\mathbf{x}/\partial\mathbf{X}^\gamma$. The first generation ($\gamma = 1$) is assumed to be present at time $t^1 = 0$, therefore its reference configuration is $\mathbf{X}^1 \equiv \mathbf{X}$ and its deformation gradient $\mathbf{F}^1 = \partial\mathbf{x}/\partial\mathbf{X}^1$ is equivalent to $\mathbf{F} = \partial\mathbf{x}/\partial\mathbf{X}$. Each generation's reference configuration $\mathbf{X}^\gamma$ has a one-to-one mapping $\mathbf{F}^{\gamma 1} = \partial\mathbf{X}^\gamma/\partial\mathbf{X}^1$ with the master reference configuration $\mathbf{X}^1$, which is that of the first generation. This mapping is postulated based on a constitutive assumption with regard to that generation's state of stress at the time of its deposition. In the current implementation, the newly deposited generation is assumed to be in a stress-free state, even though the underlying material is in a loaded configuration. Therefore, the mapping between generation γ and the first generation is simply $\mathbf{F}^{\gamma 1} = \mathbf{F}^1\left(\mathbf{X}^1, t^\gamma\right) \equiv \mathbf{F}\left(\mathbf{X}, t^\gamma\right)$. In other words, when generation γ first comes into existence, its reference configuration is the current configuration at time t^γ. Note that $\mathbf{F}^{\gamma 1}$ is a time-invariant (though not necessarily homogeneous) quantity that is determined uniquely at the birth of a generation.

The state of stress in a multigeneration solid is given by

$$\boldsymbol{\sigma} = \sum_\gamma \frac{1}{J^{\gamma 1}} \boldsymbol{\sigma}^\gamma\left(\mathbf{F}^\gamma\right)$$

where $\boldsymbol{\sigma}^\gamma\left(\mathbf{F}^\gamma\right)$ is the state of stress in the generation γ, as would be evaluated from a strain energy density function whose reference configuration is $\mathbf{X}^\gamma$. In the above equation, $J^{\gamma 1} = \det \mathbf{F}^{\gamma 1}$ and the factor $1/J^{\gamma 1}$ ensures that the strain energy density of each generation is properly normalized the volume of the material in the master reference configuration $\mathbf{X}^1$, when summing up the stresses in all the generations.

Multigeneration solids typically exhibit residual stresses when $\mathbf{F}^{\gamma 1}$ is inhomogeneous.

4.9.1 General Specification of Multigeneration Solids

The material type for a multigeneration solid is *"multigeneration"*[2]. This material describes a mixture of elastic solids, each created in a specific generation. It is a container for any combination of the elastic materials described.

<generation>	Definition of a generation.

The <generation> tag defines a new generation. It takes the following child elements.

<start_time>	"birth"-time for this generation	**[t]**
<solid>	Specification of the constitutive model for this generation	

The *solid* element defins the solid matrix constitutive relation and associated material properties.

Example:

[2] As of FEBio version 2.0, the format for defininig multi-generation materials has changed. The previous format where the generations are defined in the *Globals* section is no longer supported.

```
<material id="1" name="Growing Solid" type="multigeneration">
  <generation id="1">
    <start_time>0.0</start_time>
    <solid type="Holmes-Mow">
      <density>1</density>
      <E>1</E>
      <v>0</v>
      <beta>0.1</beta>
    </solid>
  </generation>
  <generation id="2">
    <start_time>1.0</start_time>
    <solid type="Holmes-Mow">
      <density>1</density>
      <E>1</E>
      <v>0</v>
      <beta>0.1</beta>
    </solid>
  </generation>
</material>
```

The corresponding value of t^γ for each of the generations is provided in the <Globals> section.

Example:

```
<Globals>
  <Generations>
    <gen id="1">0.0</gen>
    <gen id="2">1.0</gen>
  </Generations>
</Globals>
```

4.10 Biphasic Materials

Biphasic materials may be used to model a porous medium consisting of a mixture of a porous-permeable solid matrix and an interstitial fluid. Each of these mixture constituents is assumed to be intrinsically incompressible. This means that the true densities of the solid and fluid are invariant in space and time; this assumption further implies that a biphasic mixture will undergo zero volume change when subjected to a hydrostatic fluid pressure. Yet the mixture is compressible because the pores of the solid matrix may gain or lose fluid under general loading conditions. The Cauchy stress $\boldsymbol{\sigma}$ in a biphasic material is given by

$$\boldsymbol{\sigma} = -p\mathbf{I} + \boldsymbol{\sigma}^e \,, \tag{4.10.1}$$

where p is the interstitial fluid pressure and $\boldsymbol{\sigma}^e$ is the stress resulting from the deformation of the porous solid matrix. Therefore, the constitutive relation of the solid matrix should be chosen from the list of unconstrained materials provided in Section 4.1.4. The user is referred to the *FEBio Theory Manual* for a general description of the biphasic theory.

In addition to selecting a constitutive relation for the solid matrix, a constitutive relation must also be selected for the hydraulic permeability of the interstitial fluid flowing within the porous deformable solid matrix. The hydraulic permeability relates the volumetric flux of the fluid relative to the solid, $\mathbf{w}$, to the interstitial fluid pressure gradient, ∇p, according to

$$\mathbf{w} = -\mathbf{k} \cdot \operatorname{grad} p \,, \tag{4.10.2}$$

where $\mathbf{k}$ is the hydraulic permeability tensor. (Note that this expression does not account for the contribution of external body forces on the fluid.)

The governing equations for biphasic materials are the mixture momentum balance under quasi-static conditions, in the absence of external body force,

$$\operatorname{div} \boldsymbol{\sigma} = -\operatorname{grad} p + \operatorname{div} \boldsymbol{\sigma}^e = \mathbf{0} \,, \tag{4.10.3}$$

and the mixture mass balance,

$$\operatorname{div} (\mathbf{v}^s + \mathbf{w}) = 0 \,, \tag{4.10.4}$$

where $\mathbf{v}^s$ is the solid velocity.

4.10.1 General Specification of Biphasic Materials

The material type for a biphasic material is *"biphasic"*. The following parameters must be defined:

<solid>	Specification of the solid matrix	
<phi0>	solid volume fraction φ_r^s in the reference configuration $(0 < \varphi_r^s < 1)$	[ ]
<fluid_density>	Fluid density ρ_T^w	
<permeability>	Specification of the hydraulic permeability	
<solvent_supply>	Specification of the fluid supply $\hat{\varphi}^w$	
<tau>	Stabilization parameter τ	[**t**]

The <solid> tag encloses a description of the solid matrix constitutive relation and associated material properties, as may be selected from the list provided in Section 4.1.4. The <permeability> tag encloses a description of the permeability constitutive relation and associated material properties, as may be selected from the list presented in Section 4.10.2. The parameter <phi0> must be greater than 0 (no solid) and less than 1 (no porosity). The volume fraction of fluid (also known as the porosity) in the reference configuration is given by $1 - \varphi_r^s$. The fluid density ρ_T^w specified in <fluid_density> and the solid density ρ_T^s specified in <density> within the <solid> tag are needed only when body forces are prescribed.

Example:

```
<material id="1" name="Biphasic tissue" type="biphasic">
  <solid name="Elasticity" type="neo-Hookean">
    <E>1.0</E>
    <v>0.3</v>
  </solid>
  <phi0>0.2</phi0>
  <permeability name="Permeability" type="perm-const-iso">
    ... (description of permeability material)
  </permeability>
</material>
```

The stabilization parameter τ can be used to reduced oscillations in the fluid pressure field produced under certain loading conditions, when the finite element mesh is relatively coarse in the vicinity of a free-draining boundary (a boundary on which the fluid pressure is set to zero). When this occurs it is best to refine the mesh (create thinner elements in the direction normal to the boundary) to overcome these oscillations. When mesh refinement is not feasible or practical, one may use a non-zero stabilization parameter τ. FEBio uses an implementation based on the formulation of Aguilar et al. [1]. In our adaptation of this formulation the parameter τ has units of time. Its value is best estimated by identifying the thickness h of elements on the free-draining boundary, the characteristic modulus E of the solid matrix and hydraulic permeability k in these elements. Then τ may be set approximately to $h^2/(E \cdot k)$.

4.10.2 Permeability Materials

Permeability materials provide a constitutive relation for the hydraulic permeability of a biphasic material.

4.10.2.1 Constant Isotropic Permeability

The material type for constant isotropic permeability is "*perm-const-iso"*. The following material parameters must be defined:

<perm>	hydraulic permeability k	[**L**^{4}/**F**·**t**]

This isotropic material model uses the biphasic theory for describing the time-dependent material behavior of materials that consist of both a solid and fluid phase [60].

When the permeability is isotropic,

$$\mathbf{k} = k\,\mathbf{I}$$

For this material model, k is constant. Generally, this assumption is only reasonable when strains are small.

Example:

```
<permeability name="Permeability" type="perm-const-iso">
  <perm>0.001</perm>
</permeability>
```

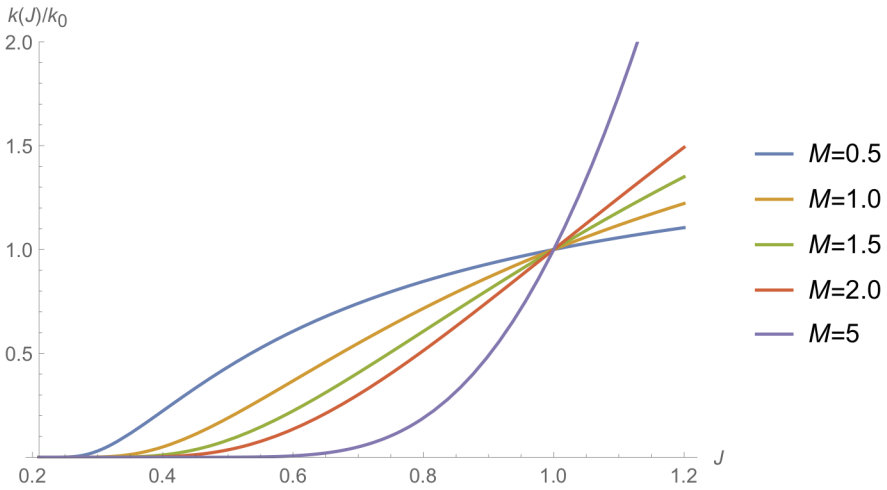


Figure 4.10.1: Exponential isotropic permeability, showing dependence of $k\left(J\right)/k_0$ on material parameter M, when $\varphi_0 = 0.2$ and $\varphi_0 < J \leq 1.2$.

4.10.2.2 Exponential Isotropic Permeability

The material type for exponential isotropic permeability is "*perm-exp-iso*". The following material parameters must be defined:

<perm>	isotropic hydraulic permeability k_0 in the reference state	[**L**^{4}/**F**·**t**]
<M>	exponential strain-dependence coefficient M $(M \geq 0)$	[ ]

This isotropic material model has the general form

$$\mathbf{k} = k\left(J\right)\mathbf{I}$$

where $J = \det \mathbf{F}$ is the determinant of the deformation gradient. For this material model,

$$k\left(J\right) = k_0 \exp\left(M\frac{J-1}{J-\varphi_0}\right),$$

where φ_0 is the referential solid volume fraction of the porous solid matrix. Pore closure occurs as J approaches φ_0 from above, in which case the permeability reduces to zero,

$$\lim_{J\to\varphi_0} k\left(J\right) = 0\,.$$

Representative variations of $k\left(J\right)/k_0$ are shown in the figure.
Example:

```
<permeability name="Permeability" type="perm-exp-iso">
  <perm>0.001</perm>
  <M>1.5</M>
</permeability>
```

4.10.2.3 Holmes-Mow

The material type for Holmes-Mow permeability is *"perm-Holmes-Mow"*. The following material parameters need to be defined:

<perm>	isotropic hydraulic permeability k_0 in the reference state	[**L**^{4}/**F**·**t**]
<M>	exponential strain-dependence coefficient M ($M \geq 0$)	[ ]
<alpha>	power-law exponent α ($\alpha \geq 0$)	[ ]

This isotropic material is similar to the constant isotropic permeability material described above, except that it uses a strain-dependent permeability tensor [42]:

$$\mathbf{k} = k\left(J\right)\mathbf{I},$$

where,

$$k\left(J\right) = k_0\left(\frac{J-\varphi_r^s}{1-\varphi_0}\right)^{\alpha} e^{\frac{1}{2}M\left(J^2-1\right)},$$

and J is the Jacobian of the deformation, i.e. $J = \det \mathbf{F}$ where $\mathbf{F}$ is the deformation gradient. Here, φ_r^s is the referential solid volume fraction in the current configuration and $\varphi_0 = \varphi_r^s\left(0\right)$ is its value in the reference configuration.

Example:

This example defines a permeability material of the Holmes-Mow type.

```
<permeability type="perm-Holmes-Mow">
  <perm>0.001</perm>
  <M>1.5</M>
  <alpha>2</alpha>
</permeability>
```

4.10.2.4 Referentially Isotropic Permeability

The material type for a biphasic material with strain-dependent permeability which is isotropic in the reference configuration is *"perm-ref-iso"*. The following material parameters need to be defined:

<perm0>	hydraulic permeability k_{0r}	[$\mathbf{L}^4/\mathbf{F}\cdot\mathbf{t}$]
<perm1>	hydraulic permeability k_{1r}	[$\mathbf{L}^4/\mathbf{F}\cdot\mathbf{t}$]
<perm2>	hydraulic permeability k_{2r}	[$\mathbf{L}^4/\mathbf{F}\cdot\mathbf{t}$]
<M>	exponential strain-dependence coefficient M ($M \geqslant 0$)	[ ]
<alpha>	power-law exponent α ($\alpha \geqslant 0$)	[ ]

This material uses a strain-dependent permeability tensor that accommodates strain-induced anisotropy:

$$\mathbf{k} = \left(k_{0r}\mathbf{I} + \frac{k_{1r}}{J^2}\mathbf{b} + \frac{k_{2r}}{J^4}\mathbf{b}^2\right)\left(\frac{J-\varphi_r^s}{1-\varphi_0}\right)^{\alpha} e^{M\left(J^2-1\right)/2},$$

where J is the Jacobian of the deformation, i.e. $J = \det \mathbf{F}$ where $\mathbf{F}$ is the deformation gradient, and $\mathbf{b} = \mathbf{F}\cdot\mathbf{F}^T$ is the left Cauchy-Green tensor. Here, φ_r^s is the referential solid volume fraction in the current configuration and $\varphi_0 = \varphi_r^s(0)$ is its value in the reference configuration.Note that the permeability in the reference state ($\mathbf{F} = \mathbf{I}$) is isotropic and given by $\mathbf{k} = (k_{0r} + k_{1r} + k_{2r})\,\mathbf{I}$.

Example:

```
<permeability name="Permeability" type="perm-ref-iso">
  <perm0>0.001</perm0>
  <perm1>0.005</perm1>
  <perm2>0.002</perm2>
  <M>1.5</M>
  <alpha>2</alpha>
</permeability>
```

4.10.2.5 Referentially Orthotropic Permeability

The material type for a poroelastic material with strain-dependent permeability which is orthotropic in the reference configuration is *"perm-ref-ortho"*. The following material parameters need to be defined:

<perm0>	isotropic hydraulic permeability k_{0r}	[**L**^{4}/**F**·**t**]
<perm1>	hydraulic permeabilities k_{1r}^a along orthogonal directions ($a = 1, 2, 3$)	[**L**^{4}/**F**·**t**]
<perm2>	hydraulic permeabilities k_{2r}^a along orthogonal directions ($a = 1, 2, 3$)	[**L**^{4}/**F**·**t**]
<M0>	isotropic exponential strain-dependence coefficient M_0 ($M_0 \geqslant 0$)	[ ]
<M>	orthotropic exponential strain-dependence coefficient M_a ($a = 1, 2, 3$, $M_a \geqslant 0$)	[ ]
<alpha0>	isotropic power-law exponent α_0 ($\alpha_0 \geqslant 0$)	[ ]
<alpha>	power-law exponent α_a ($a = 1, 2, 3$, $\alpha_a \geqslant 0$)	[ ]

This material uses a strain-dependent permeability tensor that accommodates strain-induced anisotropy:

$$\mathbf{k} = k_0\mathbf{I} + \sum_{a=1}^{3} k_1^a \mathbf{m}_a + k_2^a \left(\mathbf{m}_a \cdot \mathbf{b} + \mathbf{b} \cdot \mathbf{m}_a\right) ,$$

where,

$$\begin{aligned} k_0 &= k_{0r} \left(\frac{J - \varphi_r^s}{1 - \varphi_0}\right)^{\alpha_0} e^{M_0\left(J^2-1\right)/2} \\ k_1^a &= \frac{k_{1r}^a}{J^2} \left(\frac{J - \varphi_r^s}{1 - \varphi_0}\right)^{\alpha_a} e^{M_a\left(J^2-1\right)/2} , \quad a = 1, 2, 3 . \\ k_2^a &= \frac{k_{2r}^a}{2J^4} \left(\frac{J - \varphi_r^s}{1 - \varphi_0}\right)^{\alpha_a} e^{M_a\left(J^2-1\right)/2} \end{aligned}$$

J is the Jacobian of the deformation, i.e. $J = \det \mathbf{F}$ where $\mathbf{F}$ is the deformation gradient. $\mathbf{m}_a$ are second order tensors representing the spatial structural tensors describing the orthogonal planes of symmetry, given by

$$\mathbf{m}_a = \mathbf{F} \cdot (\mathbf{V}_a \otimes \mathbf{V}_a) \cdot \mathbf{F}^T, \quad a = 1 - 3,$$

where $\mathbf{V}_a$ are orthonormal vectors normal to the planes of symmetry (defined as described in Section 4.1.1.2). Here, φ_r^s is the referential solid volume fraction in the current configuration and $\varphi_0 = \varphi_r^s(0)$ is its value in the reference configuration. Note that the permeability in the reference state ($\mathbf{F} = \mathbf{I}$) is given by $\mathbf{k} = k_{0r}\mathbf{I} + \sum_{a=1}^{3} \left(k_{1r}^a + k_{2r}^a\right) \mathbf{V}_a \otimes \mathbf{V}_a$.

Example:

```
<permeability name="Permeability" type="perm-ref-ortho">
  <perm0>0.001</perm0>
  <perm1>0.01, 0.02, 0.03</perm1>
  <perm2>0.001, 0.002, 0.003</perm2>
  <M0>0.5</M0>
  <M>1.5, 2.0, 2.5</M>
```

```
    <alpha0>1.5</alpha0>
    <alpha>2, 2.5, 3</alpha>
  </permeability>
```

4.10.2.6 Referentially Transversely Isotropic Permeability

The material type for a biphasic material with strain-dependent permeability which is transversely isotropic in the reference configuration is *"perm-ref-trans-iso"*. The following material parameters need to be defined:

<perm0>	isotropic hydraulic permeability k_{0r}	[**L**^{4}/**F**·**t**]
<perm1A>	axial hydraulic permeability k_{1r}^A	[**L**^{4}/**F**·**t**]
<perm2A>	axial hydraulic permeability k_{2r}^A	[**L**^{4}/**F**·**t**]
<perm1T>	transverse hydraulic permeability k_{1r}^T	[**L**^{4}/**F**·**t**]
<perm2T>	transverse hydraulic permeability k_{2r}^T	[**L**^{4}/**F**·**t**]
<M0>	isotropic exponential strain-dependence coefficient M_0 ($M_0 \geqslant 0$)	[ ]
<MA>	axial exponential strain-dependence coefficient M_A ($M_A \geqslant 0$)	[ ]
<MT>	transverse exponential strain-dependence coefficient M_T ($M_T \geqslant 0$)	[ ]
<alpha0>	isotropic power-law exponent α_0 ($\alpha_0 \geqslant 0$)	[ ]
<alphaA>	axial power-law exponent α_A ($\alpha_A \geqslant 0$)	[ ]
<alphaT>	transverse power-law exponent α_T ($\alpha_T \geqslant 0$)	[ ]

This material uses a strain-dependent permeability tensor that accommodates strain-induced anisotropy:

$$\begin{aligned}\mathbf{k} = & k_{0r}\left(\frac{J-\varphi_r^s}{1-\varphi_0}\right)^{\alpha_0} e^{M_0\left(J^2-1\right)/2}\mathbf{I} \\ & + \left(\frac{k_{1r}^T}{J^2}\left(\mathbf{b}-\mathbf{m}\right) + \frac{k_{2r}^T}{2J^4}\left[2\mathbf{b}^2 - \left(\mathbf{m}\cdot\mathbf{b}+\mathbf{b}\cdot\mathbf{m}\right)\right]\right)\left(\frac{J-\varphi_r^s}{1-\varphi_0}\right)^{\alpha_T} e^{M_T\left(J^2-1\right)/2}, \\ & + \left(\frac{1}{J^2}k_{1r}^A\mathbf{m} + \frac{1}{2J^4}k_{2r}^A\left(\mathbf{m}\cdot\mathbf{b}+\mathbf{b}\cdot\mathbf{m}\right)\right)\left(\frac{J-\varphi_r^s}{1-\varphi_0}\right)^{\alpha_A} e^{M_A\left(J^2-1\right)/2}\end{aligned}$$

where J is the Jacobian of the deformation, i.e. $J = \det\mathbf{F}$ where $\mathbf{F}$ is the deformation gradient, and $\mathbf{b} = \mathbf{F}\cdot\mathbf{F}^T$ is the left Cauchy-Green tensor. $\mathbf{m}$ is a second order tensor representing the spatial structural tensor describing the axial direction, given by

$$\mathbf{m} = \mathbf{F}\cdot\left(\mathbf{V}\otimes\mathbf{V}\right)\cdot\mathbf{F}^T,$$

where $\mathbf{V}$ is a unit vector along the axial direction (defined as described in Section 4.1.1). Here, φ_r^s is the referential solid volume fraction in the current configuration and $\varphi_0 = \varphi_r^s\left(0\right)$ is its value in the reference configuration. Note that the permeability in the reference state ($\mathbf{F} = \mathbf{I}$) is given by,

$$\mathbf{k}_{|\mathbf{F}=\mathbf{I}} = \left(k_{0r} + k_{1r}^T + k_{2r}^T\right)\mathbf{I} + \left(k_{1r}^A - k_{1r}^T + k_{2r}^A - k_{2r}^T\right)\left(\mathbf{V}\otimes\mathbf{V}\right)$$

Example:

```
<permeability name="Permeability" type="perm-ref-trans-iso">
  <perm0>0.002</perm0>
  <perm1A>0.01</perm1A>
  <perm2A>0.01</perm2A>
  <perm1T>0.001</perm1T>
  <perm2T>0.05</perm2T>
  <M0>1.0</M0>
```

```
      <MA>0.5</MA>
      <MT>1.5</MT>
      <alpha0>1.0</alpha0>
      <alphaA>0.5</alphaA>
      <alphaT>2.0</alphaT>
    </permeability>
```

4.10.3 Fluid Supply Materials

Fluid supply materials may be used to simulate an external source of fluid, such as supply from microvasculature that is not modeled explicitly. The fluid supply term, $\hat{\varphi}^w$, appears in the mass balance relation for the mixture,

$$\operatorname{div}\left(\mathbf{v}^s + \mathbf{w}\right) = \hat{\varphi}^w \,.$$

$\hat{\varphi}^w$ has units of reciprocal time [$\mathbf{t}^{-1}$]; it represents the rate at which the volume fraction of fluid changes with time.

4.10.3.1 Starling Equation

The material type for Starling's equation for fluid supply is *"Starling"*. The following material parameters need to be defined:

<kp>	hydraulic filtration coefficient, k_p	[**L**^{2}/**F**·**t**]
<pv>	fluid pressure in external source, p_v	[**P**]

The fluid supply is given by Starling's equation,

$$\hat{\varphi}^w = k_p \left(p_v - p \right) ,$$

where p is the fluid pressure in the biphasic material.

Example:

This example defines a fluid supply material of the Starling type.

```
<solvent_supply type="Starling">
  <kp>0.001</kp>
  <pv>0.1</pv>
</solvent_supply>
```

4.11 Biphasic-Solute Materials

Biphasic-solute materials may be used to model the transport of a solvent and a solute in a neutral porous solid matrix, under isothermal conditions. Each of these mixture constituents is assumed to be intrinsically incompressible. This means that their true densities are invariant in space and time; this assumption further implies that a biphasic-solute mixture will undergo zero volume change when subjected to a hydrostatic fluid pressure. Yet the mixture is compressible because the pores of the solid matrix may gain or lose fluid under general loading conditions. Therefore, the constitutive relation of the solid matrix should be chosen from the list of unconstrained materials provided in Section 4.1.4. The volume fraction of the solute is assumed to be negligible relative to the volume fractions of the solid or solvent. This means that the mixture will not change in volume as the solute concentration changes. As the solute transports through the mixture, it may experience frictional interactions with the solvent and the solid. This friction may act as a hindrance to the solute transport, or may help convect the solute through the mixture. The distinction between frictional exchanges with the solvent and solid is embodied in the specification of two diffusivities for the solute: The free diffusivity, which represents diffusivity in the absence of a solid matrix (frictional exchange only with the solvent) and the mixture diffusivity, which represents the combined frictional interactions with the solvent and solid matrix. The user is referred to the *FEBio Theory Manual* for a more detailed description of the biphasic-solute theory.

Due to steric volume exclusion and short-range electrostatic interactions, the solute may not have access to all of the pores of the solid matrix. In other words, only a fraction κ of the pores is able to accommodate a solute of a particular size ($0 < \kappa \leqslant 1$). Furthermore, the activity γ of the solute (the extent by which the solute concentration influences its chemical potential) may be similarly altered by the surrounding porous solid matrix. Therefore, the combined effects of volume exclusion and attenuation of activity may be represented by the effective solubility $\tilde{\kappa} = \kappa/\gamma$, such that the chemical potential μ of the solute is given by

$$\mu = \mu_0(\theta) + \frac{R\theta}{M} \ln \frac{c}{\tilde{\kappa}} . \tag{4.11.1}$$

In this expression, μ_0 is the solute chemical potential at some reference temperature θ; c is the solute concentration on a solution-volume basis (number of moles of solute per volume of interstitial fluid in the mixture); M is the solute molecular weight (an invariant quantity); and R is the universal gas constant. In a biphasic-solute material, a constitutive relation is needed for $\tilde{\kappa}$; in general, $\tilde{\kappa}$ may be a function of the solid matrix strain and the solute concentration. In FEBio, the dependence of the effective solubility on the solid matrix strain is currently constrained to a dependence on $J = \det \mathbf{F}$.

In a biphasic-solute material, the interstitial fluid pressure p is influenced by both mechanical and chemical environments. In other words, this pressure includes both mechanical and osmotic contributions, the latter arising from the presence of the solute. The solvent mechano-chemical potential $\tilde{\mu}^w$ is given by

$$\tilde{\mu}^w = \mu_0^w(\theta) + \frac{1}{\rho_T^w}(p - R\theta\Phi c) , \tag{4.11.2}$$

where μ_0^w is the solvent chemical potential at some reference temperature θ; ρ_T^w is the true density of the solvent (an invariant property for an intrinsically incompressible fluid); and Φ is the osmotic coefficient which represents the extent by which the solute concentration influences the solvent chemical potential. In a biphasic-solute material, a constitutive relation is needed for Φ; in general, Φ may be a function of the solid matrix strain and the solute concentration. In FEBio, the depen-

dence of the osmotic coefficient on the solid matrix strain is currently constrained to a dependence on $J = \det \mathbf{F}$.

The solute mechano-chemical potential is nearly equal to its chemical potential because the solute volume fraction is assumed to be negligible. In general, momentum and energy balances evaluated across a boundary surface in a biphasic-solute mixture require that the mechano-chemical potentials of solvent and solute be continuous across that surface. These continuity requirements are enforced automatically in FEBio by defining the effective fluid pressure $\tilde{p}$ and solute concentration $\tilde{c}$ as

$$\begin{aligned} \tilde{p} &= p - R\theta\Phi c \\ \tilde{c} &= \frac{c}{\tilde{\kappa}} \end{aligned} \quad . \tag{4.11.3}$$

Therefore, nodal variables in FEBio consist of the solid matrix displacement $\mathbf{u}$, the effective fluid pressure $\tilde{p}$, and the effective solute concentration $\tilde{c}$. Essential boundary conditions must be imposed on these variables, and not on the actual pressure p or concentration c. (In a biphasic material however, since $c = 0$, the effective and actual fluid pressures are the same, $p = \tilde{p}$.)

The mixture stress in a biphasic-solute material is given by $\boldsymbol{\sigma} = -p\mathbf{I} + \boldsymbol{\sigma}^e$, where $\boldsymbol{\sigma}^e$ is the stress arising from the solid matrix strain. The mixture traction on a surface with unit outward normal $\mathbf{n}$ is $\mathbf{t} = \boldsymbol{\sigma} \cdot \mathbf{n}$. This traction is continuous across the boundary surface. Therefore, the corresponding natural boundary condition for a biphasic-solute mixture is $\mathbf{t} = \mathbf{0}$. (In other words, if no boundary condition is imposed on the solid matrix displacement or mixture traction, the natural boundary condition is in effect.)

The natural boundary conditions for the solvent and solute are similarly $\mathbf{w} \cdot \mathbf{n} = 0$ and $\mathbf{j} \cdot \mathbf{n} = 0$, where $\mathbf{w}$ is the volumetric flux of solvent relative to the solid and $\mathbf{j}$ is the molar flux of solute relative to the solid. In general, $\mathbf{w}$ and $\mathbf{j}$ are given by

$$\begin{aligned} \mathbf{w} &= -\tilde{\mathbf{k}} \cdot \left(\nabla\tilde{p} + R\theta \frac{\tilde{\kappa}}{d_0} \mathbf{d} \cdot \nabla\tilde{c} \right) \\ \mathbf{j} &= \tilde{\kappa}\mathbf{d} \cdot \left(-\varphi^w \nabla\tilde{c} + \frac{\tilde{c}}{d_0}\mathbf{w} \right) \end{aligned} \quad , \tag{4.11.4}$$

where

$$\tilde{\mathbf{k}} = \left[\mathbf{k}^{-1} + \frac{R\theta}{\varphi^w} \frac{\tilde{\kappa} c}{d_0} \left(\mathbf{I} - \frac{\mathbf{d}}{d_0} \right) \right]^{-1} \tag{4.11.5}$$

is the effective hydraulic permeability of the interstitial fluid solution (solvent and solute) through the porous solid matrix; $\mathbf{k}$ is the hydraulic permeability of the solvent through the porous solid matrix; $\mathbf{d}$ is the solute diffusivity through the mixture (frictional interactions with solvent and solid); and d_0 is the solute free diffusivity (frictional interactions with solvent only). $\varphi^w \approx 1 - \varphi^s$ is the solid matrix porosity in the current configuration. The above expressions for the solvent and solute flux do not account for external body forces.

The governing equations for a biphasic-solute material are the momentum balance for the mixture, Eq.(4.10.3), the mass balance for the mixture, which reduces to Eq.(4.10.4) under the assumption of dilute solutions, and the mass balance for the solute,

$$\frac{\partial \left(\varphi^w c \right)}{\partial t} + \operatorname{div} \left(\mathbf{j} + \varphi^w c \mathbf{v}^s \right) = 0 \, . \tag{4.11.6}$$

4.11.1 Guidelines for Biphasic-Solute Analyses

4.11.1.1 Prescribed Boundary Conditions

In most analyses, it may be assumed that the ambient fluid pressure in the external environment is zero, thus $p_* = 0$, where the subscripted asterisk is used to denote environmental conditions. The ambient solute concentration may be represented by c_*. It follows that the effective fluid pressure in the external environment is $\tilde{p}_* = -R\theta\Phi_* c_*$ and the effective concentration is $\tilde{c}_* = c_*/\tilde{\kappa}_*$. Therefore, in biphasic-solute analyses, whenever the external environment contains a solute at a concentration of c_*, the user must remember to prescribe non-zero boundary conditions for the effective solute concentration *and* the effective fluid pressure.

Letting $p_* = 0$ also implies that prescribed mixture normal tractions (Section 3.13.2.5) represent only the traction above ambient conditions. Note that users are not obligated to assume that $p_* = 0$. However, if a non-zero value is assumed for the ambient pressure, then users must remember to incorporate this non-zero value whenever prescribing mixture normal tractions.

4.11.1.2 Prescribed Initial Conditions

When a biphasic-solute material is initially exposed to a given external environment with effective pressure $\tilde{p}_*$ and effective concentration $\tilde{c}_*$, the initial conditions inside the material should be set to $\tilde{p} = \tilde{p}_*$ and $\tilde{c} = \tilde{c}_*$ in order to produce the correct initial state. The values of $\tilde{p}_*$ and $\tilde{c}_*$ should be evaluated as described in Section 8.7.2.

4.11.2 General Specification of Biphasic-Solute Materials

The material type for a biphasic-solute material is *"biphasic-solute"*. Constitutive relations must be provided for the solid matrix, the hydraulic permeability $\mathbf{k}$, the solute diffusivities $\mathbf{d}$ and d_0, the effective solubility $\tilde{\kappa}$ and the osmotic coefficient Φ. Therefore, the following parameters must be defined:

<solid>	specification of the solid matrix
<phi0>	solid volume fraction φ_r^s in the reference configuration
<permeability>	specification of the hydraulic permeability $\mathbf{k}$
<osmotic_coefficient>	specification of the osmotic coefficient $\Phi \quad \mathbf{d}d_0$
<solute>	specification of the solute properties

The <solid> tag encloses a description of the solid matrix constitutive relation and associated material properties, as may be selected from the list provided in Section 4.1.4. The solid volume fraction in the reference configuration, <phi0>, must be greater than 0 (no solid) and less than 1 (only solid). The volume fraction of fluid (also known as the porosity) in the reference configuration is given by $1 - \varphi_0$. The <permeability> tag encloses a description of the permeability constitutive relation and associated material properties, as may be selected from the list presented in Section 4.10.2.

The <solute> tag provides a description of the solute in the biphasic-solute mixture. This tag includes the required *sol* attribute, which should reference a solute *id* from the <Solutes> description in the <Globals> section (Section 3.4.2). The following parameters must be defined in this description:

<diffusivity>	specification of the solute diffusivities $\mathbf{d}$ and d_0
<solubility>	specification of the solute effective solubility $\tilde{\kappa}$

The <diffusivity> and <solubility> tags enclose descriptions of materials that may be selected from the lists presented in Sections 4.11.3 and 4.11.4, respectively. Each solute tag must include a "sol" attribute.

Example:

```
<material id="1" name="Biological tissue" type="biphasic-solute">
  <solid name="Elasticity" type="neo-Hookean">
    <E>1.0</E>
    <v>0.3</v>
  </solid>
  <phi0>0.2</phi0>
  <permeability name="Permeability" type="perm-const-iso">
    ... (description of permeability material)
  </permeability>
  <osmotic_coefficient name="Osmotic" type="osm-coef-const">
    ... (description of osmotic coefficient material)
  </osmotic_coefficient>
  <solute sol="1">
    <diffusivity name="Diffusivity" type="diff-const-iso">
      ... (description of diffusivity material)
```

```
      </diffusivity>
      <solubility name="Solubility" type="solub-const">
        ... (description of solubility material)
      </solubility>
    </solute>
  </material>
```

When a biphasic-solute material is employed in an analysis, it is also necessary to specify the values of the universal gas constant R [**F**·**L**/**n**·**T**] and absolute temperature θ [**T**] under <Constants> in the <Globals> section, using a self-consistent set of units. A solute must also be defined in the <Solutes> section, whose id should be associated with the "sol" attribute in the solute material description.

Example:

```
<Globals>
  <Constants>
    <R>8.314</R>
    <T>298</T>
  </Constants>
  <Solutes>
    <solute id="1" name="neutral">
      <charge_number>0</charge_number>
    </solute>
  </Solutes>
</Globals>
```

It is also possible to create models with biphasic-solute materials that use different solutes in different regions. In that case, introduce additional solute entries in the <Solutes> section and refer to those solute ids in the biphasic-solute material descriptions.

4.11.3 Diffusivity Materials

Diffusivity materials provide a constitutive relation for the solute diffusivity in a biphasic-solute material. In general, the diffusivity tensor $\mathbf{d}$ may be a function of strain and solute concentration.

4.11.3.1 Constant Isotropic Diffusivity

The material type for constant isotropic diffusivity materials is "*diff-const-iso"*. The following material parameters must be defined:

<free_diff>	free diffusivity d_0 of solute (diffusivity in solution)	[**L**^{2}/**t**]
<diff>	constant isotropic diffusivity d of solute in biphasic-solute mixture	[**L**^{2}/**t**]

When the permeability is isotropic,

$$\mathbf{d} = d\mathbf{I}$$

For this material model, d is constant. This assumption is only true when strains are small. Note that the user must specify $d \leqslant d_0$, since a solute cannot diffuse through the biphasic-solute mixture faster than in free solution.

Example:

```
<diffusivity name="Permeability" type="diff-const-iso">
  <free_diff>1e-9</ free_diff >
  <diff>0.5e-9</diff>
</diffusivity>
```

4.11.3.2 Constant Orthotropic Diffusivity

The material type for constant orthotropic diffusivity materials is "*diff-const-ortho*". The following material parameters must be defined:

<free_diff>	free diffusivity d_0 of solute (diffusivity in solution)	[**L**^{2}/**t**]
<diff>	diffusivities d^a along orthogonal directions $(a = 1, 2, 3)$	[**L**^{2}/**t**]

When the permeability is orhotropic,

$$\mathbf{d} = \sum_{a=1}^{3} d^a \mathbf{V}_a \otimes \mathbf{V}_a$$

where $\mathbf{V}_a$ are orthonormal vectors normal to the planes of symmetry (defined as described in Section 4.1.1). For this material model, d^a's are constant. Therefore this model should be used only when strains are small. Note that the user must specify $d^a \leqslant d_0$, since a solute cannot diffuse through the biphasic-solute mixture faster than in free solution.

Example:

```
<diffusivity name="Permeability" type="diff-const-ortho">
  <free_diff>0.005</free_diff >
  <diff>0.002,0.001,0.003</diff>
</diffusivity>
```

4.11.3.3 Referentially Isotropic Diffusivity

The material type for a strain-dependent diffusivity tensor which is isotropic in the reference configuration is *"diff-ref-iso"*. The following material parameters need to be defined:

<free_diff>	free diffusivity d_0	[**L**^{2}/**t**]
<diff0>	diffusivity d_{0r}	[**L**^{2}/**t**]
<diff1>	diffusivity d_{1r}	[**L**^{2}/**t**]
<diff2>	diffusivity d_{2r}	[**L**^{2}/**t**]
<M>	exponential strain-dependence coefficient M ($M \geqslant 0$)	[ ]
<alpha>	power-law exponent α ($\alpha \geqslant 0$)	[ ]

This material uses a strain-dependent diffusivity tensor that accommodates strain-induced anisotropy:

$$\mathbf{d} = \left(d_{0r}\mathbf{I} + \frac{d_{1r}}{J^2}\mathbf{b} + \frac{d_{2r}}{J^4}\mathbf{b}^2 \right) \left(\frac{J - \varphi_r^s}{1 - \varphi_r^s} \right) e^{M\left(J^2-1\right)/2},$$

where $J$is the jacobian of the deformation, i.e. $J = \det \mathbf{F}$ where $\mathbf{F}$ is the deformation gradient, and $\mathbf{b} = \mathbf{F} \cdot \mathbf{F}^T$ is the left Cauchy-Green tensor. Note that the diffusivity in the reference state ($\mathbf{F} = \mathbf{I}$) is isotropic and given by $\mathbf{d} = (d_{0r} + d_{1r} + d_{2r})\,\mathbf{I}$.

Example:

```
<diffusivity name="Diffusivity" type="diff-ref-iso">
  <phi0>0.2</phi0>
  <free_diff>0.005</free_diff>
  <diff0>0.001</diff0>
  <diff1>0.005</diff1>
  <diff2>0.002</diff2>
  <M>1.5</M>
  <alpha>2</alpha>
</diffusivity>
```

4.11.3.4 Referentially Orthotropic Diffusivity

The material type for a strain-dependent diffusivity which is orthotropic in the reference configuration is *"diff-ref-ortho"*. The following material parameters need to be defined:

<free_diff>	free diffusivity d_0	[$\mathbf{L}^2$/**t**]
<diff0>	isotropic diffusivity d_{0r}	[$\mathbf{L}^2$/**t**]
<diff1>	diffusivities d^a_{1r} along orthogonal directions ($a = 1, 2, 3$)	[$\mathbf{L}^2$/**t**]
<diff2>	diffusivities d^a_{2r} along orthogonal directions ($a = 1, 2, 3$)	[$\mathbf{L}^2$/**t**]
<M0>	isotropic exponential strain-dependence coefficient M_0 ($M_0 \geqslant 0$)	[ ]
<M>	orthotropic exponential strain-dependence coefficient M_a ($a = 1, 2, 3$, $M_a \geqslant 0$)	[ ]
<alpha0>	isotropic power-law exponent α_0 ($\alpha_0 \geqslant 0$)	[ ]
<alpha>	power-law exponent α_a ($a = 1, 2, 3$, $\alpha_a \geqslant 0$)	[ ]

This material uses a strain-dependent diffusivity tensor that accommodates strain-induced anisotropy:

$$\mathbf{d} = d_0\mathbf{I} + \sum_{a=1}^{3} d^a_1\mathbf{m}_a + d^a_2\left(\mathbf{m}_a\cdot\mathbf{b} + \mathbf{b}\cdot\mathbf{m}_a\right),$$

where,

$$d_0 = d_{0r}\left(\frac{J-\varphi^s_r}{1-\varphi^s_r}\right)^{\alpha_0} e^{M_0\left(J^2-1\right)/2}$$
$$d^a_1 = \frac{d^a_{1r}}{J^2}\left(\frac{J-\varphi^s_r}{1-\varphi^s_r}\right)^{\alpha_a} e^{M_a\left(J^2-1\right)/2}, \quad a = 1, 2, 3$$
$$d^a_2 = \frac{d^a_{2r}}{2J^4}\left(\frac{J-\varphi^s_r}{1-\varphi^s_r}\right)^{\alpha_a} e^{M_a\left(J^2-1\right)/2}$$

J is the Jacobian of the deformation, i.e. $J = \det\mathbf{F}$ where $\mathbf{F}$ is the deformation gradient. $\mathbf{m}_a$ are second order tensors representing the spatial structural tensors describing the orthogonal planes of symmetry, given by

$$\mathbf{m}_a = \mathbf{F}\cdot\left(\mathbf{V}_a\otimes\mathbf{V}_a\right)\cdot\mathbf{F}^T, \quad a = 1-3,$$

where $\mathbf{V}_a$ are orthonormal vectors normal to the planes of symmetry (defined as described in Section 4.1.1). Note that the diffusivity in the reference state ($\mathbf{F} = \mathbf{I}$) is given by $\mathbf{d} = d_{0r}\mathbf{I} + \sum_{a=1}^{3}\left(d^a_{1r} + d^a_{2r}\right)\mathbf{V}_a\otimes\mathbf{V}_a$.

Example:

```
<diffusivity name="Diffusivity" type="diff-ref-ortho">
  <phi0>0.2</phi0>
  <free_diff>0.005</free_diff>
  <diff00>0.001</diff00>
  <diff1>0.01, 0.02, 0.03</diff1>
  <diff2>0.001, 0.002, 0.003</diff2>
  <M0>0.5</M0>
```

```
      <M>1.5, 2.0, 2.5</M>
      <alpha0>1.5</alpha0>
      <alpha>2, 2.5, 3</alpha>
    </diffusivity>
```

4.11.3.5 Albro Isotropic Diffusivity

The material type for a porosity and concentration-dependent diffusivity which is isotropic is *"diff-Albro-iso"*. The following material parameters need to be defined:

<free_diff>	free diffusivity d_0	[**L**^{2}/**t**]
<cdinv>	inverse of characteristic concentration for concentration-dependence c_D^{-1}	[**L**^{3}/**n**]
<alphad>	coefficient of porosity-dependence α_D	[ ]

This material uses a porosity and concentration-dependent diffusivity tensor that remains isotropic with deformation:

$$\mathbf{d} = d_0 \exp\left(-\alpha_D \frac{1-\varphi^w}{\varphi^w} - \frac{c}{c_D}\right)\mathbf{I},$$

where $c_D^{-1} = 1/c_D$ and the porosity φ^w varies with deformation according to the kinematic constraint

$$\varphi^w = 1 - \frac{\varphi_r^s}{J}\,.$$

J is the Jacobian of the deformation, i.e. $J = \det \mathbf{F}$ where $\mathbf{F}$ is the deformation gradient and φ_r^s is the referential solid volume fraction. Here, c represents the actual concentration of the solute whose diffusivity is given by $\mathbf{d}$. This constitutive relation is based on the experimental findings reported by Albro et al. [2].

Example:

```
<solute sol="1">
  <diffusivity type="diff-Albro-iso">
    <free_diff>123e-6</free_diff>
    <alphad>18</alphad>
    <cdinv>0.0625</cdinv>
  </diffusivity>
  <solubility type="solub-const">
    <solub>1</solub>
  </solubility>
</solute>
```

4.11.4 Solubility Materials

4.11.4.1 Constant Solubility

The material type for constant solubility materials is "*solub-const*". The following material parameters must be defined:

<solub>	solubility $\tilde{\kappa}$	[ ]

For this material model, $\tilde{\kappa}$ is constant.
Example:

```
<solubility name="Solubility" type="solub-const">
  <solub>1</solub>
</solubility>
```

4.11.5 Osmotic Coefficient Materials

4.11.5.1 Constant Osmotic Coefficient

The material type for constant osmotic coefficient materials is "*osm-coef-const*". The following material parameters must be defined:

<osmcoef>	Osmotic coefficient Φ	[ ]

For this material model, Φ is constant.
Example:

```
<osmotic_coefficient name="Osmotic coefficient" type="osm-coef-const">
  <osmcoef>1</osmcoef>
</osmotic_coefficient>
```

4.12 Triphasic and Multiphasic Materials

Triphasic materials may be used to model the transport of a solvent and a pair of monovalent salt counter-ions (two solutes with charge numbers $+1$ and -1) in a charged porous solid matrix, under isothermal conditions. Multiphasic materials may be used to model the transport of a solvent and any number of neutral or charged solutes; a triphasic mixture is a special case of a multiphasic mixture. Each of the mixture constituents is assumed to be intrinsically incompressible. This means that their true densities are invariant in space and time; this assumption further implies that a multiphasic mixture will undergo zero volume change when subjected to a hydrostatic fluid pressure. Yet the mixture is compressible because the pores of the solid matrix may gain or lose fluid under general loading conditions. Therefore, the constitutive relation of the solid matrix should be chosen from the list of unconstrained materials provided in Section 4.1.4. The volume fraction of the solutes is assumed to be negligible relative to the volume fractions of the solid or solvent. This means that the mixture will not change in volume as the solute concentrations change. As the solutes transport through the mixture, they may experience frictional interactions with the solvent and the solid. This friction may act as a hindrance to the solute transport, or may help convect the solutes through the mixture. The distinction between frictional exchanges with the solvent and solid is embodied in the specification of two diffusivities for each solute: The free diffusivity, which represents diffusivity in the absence of a solid matrix (frictional exchange only with the solvent) and the mixture diffusivity, which represents the combined frictional interactions with the solvent and solid matrix. The user is referred to the *FEBio Theory Manual* for a more detailed description of triphasic and multiphasic theory.

Due to steric volume exclusion and short-range electrostatic interactions, a solute α may not have access to all of the pores of the solid matrix. In other words, only a fraction κ^α of the pores is able to accommodate solute α ($0 < \kappa^\alpha \leqslant 1$). Furthermore, the activity γ^α of solute α (the extent by which the solute concentration influences its chemical potential) may be similarly altered by the surrounding porous solid matrix. Therefore, the combined effects of volume exclusion and attenuation of activity may be represented by the effective solubility $\hat{\kappa}^\alpha = \kappa^\alpha/\gamma^\alpha$, such that the chemical potential μ of the solute is given by

$$\mu^\alpha = \mu_0^\alpha(\theta) + \frac{R\theta}{M^\alpha}\ln\frac{c^\alpha}{\hat{\kappa}^\alpha}\,. \tag{4.12.1}$$

In this expression, μ_0^α is the solute chemical potential at some reference temperature θ; c^α is the solute concentration on a solution-volume basis (number of moles of solute per volume of interstitial fluid in the mixture); M^α is the solute molecular weight (an invariant quantity); and R is the universal gas constant. In a triphasic material, a constitutive relation is needed for $\hat{\kappa}^\alpha$; in general, $\hat{\kappa}^\alpha$ may be a function of the solid matrix strain and the solute concentration. In FEBio, the dependence of the effective solubility on the solid matrix strain is currently constrained to a dependence on $J = \det\mathbf{F}$.

The solid matrix of a multiphasic material may be charged and its charge density is given by c^F. This charge density may be either negative or positive. The charge density varies with the deformation, increasing when the pore volume decreases. Based on the balance of mass for the solid,

$$c^F = \frac{1-\varphi_r^s}{J-\varphi_r^s}c_r^F\,, \tag{4.12.2}$$

where φ_r^s is the solid volume fraction and c_r^F is the fixed charge density in the reference configuration.

In the multiphasic theory it is assumed that electroneutrality is satisfied at all times. In other words, the net charge of the mixture is always zero (neutral). This electroneutrality condition is represented by a constraint equation on the ion concentrations,

$$c^F + \sum_\alpha z^\alpha c^\alpha = 0\,, \tag{4.12.3}$$

where z^α is the charge number of solute $\alpha z^- = -1$. Since the concentrations of the cation and anion inside the triphasic material are not the same, an electrical potential difference is produced between the interstitial and external environments. The electric potential in the triphasic mixture is denoted by ψ and its effect combines with the chemical potential of each solute to produce the electrochemical potential $\tilde{\mu}^\alpha$, where

$$\tilde{\mu}^\alpha = \mu_0^\alpha(\theta) + \frac{R\theta}{M^\alpha}\ln\frac{c^\alpha}{\hat{\kappa}^\alpha} + \frac{z^\alpha}{M^\alpha}F_c\psi\,. \tag{4.12.4}$$

In this expression, F_c represents Faraday's constant. It is also possible to rearrange this expression as

$$\tilde{\mu}^\alpha = \mu_0^\alpha(\theta) + \frac{R\theta}{M^\alpha}\ln\left[\exp\left(z^\alpha\frac{F_c\psi}{R\theta}\right)\frac{c^\alpha}{\hat{\kappa}^\alpha}\right]\,. \tag{4.12.5}$$

In a multiphasic material, the interstitial fluid pressure p is influenced by both mechanical and chemical environments. In other words, this pressure includes both mechanical and osmotic contributions, the latter arising from the presence of the solutes. The solvent mechano-chemical potential $\tilde{\mu}^w$ is given by

$$\tilde{\mu}^w = \mu_0^w(\theta) + \frac{1}{\rho_T^w}\left(p - R\theta\Phi\sum_\alpha c^\alpha\right)\,, \tag{4.12.6}$$

where μ_0^w is the solvent chemical potential at some reference temperature θ; ρ_T^w is the true density of the solvent (an invariant property for an intrinsically incompressible fluid); and Φ is the osmotic coefficient which represents the extent by which the solute concentrations influence the solvent chemical potential. In a multiphasic material, a constitutive relation is needed for Φ; in general, Φ may be a function of the solid matrix strain and the solute concentrations. In FEBio, the dependence of the osmotic coefficient on the solid matrix strain is currently constrained to a dependence on $J = \det\mathbf{F}$.

The solute mechano-electrochemical potential is nearly equal to its electrochemical potential because the solute volume fraction is assumed to be negligible. The solvent mechano-electrochemical potential is the same as its mechano-chemical potential, since the solvent is neutral in a multiphasic mixture. In general, momentum and energy balances evaluated across a boundary surface in a multiphasic mixture require that the mechano-electrochemical potentials of solvent and solutes be continuous across that surface. These continuity requirements are enforced automatically in FEBio by defining the effective fluid pressure $\tilde{p}$ and solute concentration $\tilde{c}^\alpha$ as

$$\begin{aligned}\tilde{p} &= p - R\theta\Phi\textstyle\sum_\alpha c^\alpha\\ \tilde{c}^\alpha &= c^\alpha/\tilde{\kappa}^\alpha\end{aligned}\,, \tag{4.12.7}$$

where

$$\tilde{\kappa}^\alpha = \hat{\kappa}^\alpha\exp\left(-z^\alpha\frac{F_c\psi}{R\theta}\right) \tag{4.12.8}$$

is the partition coefficient for solute α. The partition coefficient incorporates the combined effects of solubility and long-range electrostatic interactions to determine the ratio of interstitial to external concentration for that solute. Therefore, the effective concentration represents a measure of the *activity* of the solute, as understood in chemistry. The effective fluid pressure represents that part of the pressure which is over and above osmotic effects.

In FEBio, nodal variables consist of the solid matrix displacement $\mathbf{u}$, the effective fluid pressure $\tilde{p}$, and the effective solute concentrations $\tilde{c}^\alpha$. Essential boundary conditions must be imposed on these variables, and not on the actual pressure p or concentrations c^α. (In a biphasic material however, since $c^\alpha = 0$, the effective and actual fluid pressures are the same, $p = \tilde{p}$.)

The mixture stress in a triphasic material is given by $\boldsymbol{\sigma} = -p\mathbf{I} + \boldsymbol{\sigma}^e$, where $\boldsymbol{\sigma}^e$ is the stress arising from the solid matrix strain. The mixture traction on a surface with unit outward normal $\mathbf{n}$ is $\mathbf{t} = \boldsymbol{\sigma} \cdot \mathbf{n}$. This traction is continuous across the boundary surface. Therefore, the corresponding natural boundary condition for a multiphasic mixture is $\mathbf{t} = \mathbf{0}$. (In other words, if no boundary condition is imposed on the solid matrix displacement or mixture traction, the natural boundary condition is in effect.)

The natural boundary conditions for the solvent and solutes are similarly $\mathbf{w}\cdot\mathbf{n} = 0$ and $\mathbf{j}^\alpha\cdot\mathbf{n} = 0$, where $\mathbf{w}$ is the volumetric flux of solvent relative to the solid and $\mathbf{j}^\alpha$ is the molar flux of solute α relative to the solid. In general, $\mathbf{w}$ and $\mathbf{j}^\alpha$ are given by

$$\begin{aligned} \mathbf{w} &= -\tilde{\mathbf{k}} \cdot \left(\nabla\tilde{p} + R\theta \sum_\alpha \frac{\tilde{\kappa}^\alpha}{d_0^\alpha} \mathbf{d}^\alpha \cdot \nabla\tilde{c}^\alpha \right), \\ \mathbf{j}^\alpha &= \tilde{\kappa}^\alpha\, \mathbf{d}^\alpha \cdot \left(-\varphi^w \nabla\tilde{c}^\alpha + \frac{\tilde{c}^\alpha}{d_0^\alpha}\mathbf{w} \right), \end{aligned} \tag{4.12.9}$$

where

$$\tilde{\mathbf{k}} = \left[\mathbf{k}^{-1} + \frac{R\theta}{\varphi^w} \sum_\alpha \frac{\tilde{\kappa}^\alpha c^\alpha}{d_0^\alpha} \left(\mathbf{I} - \frac{\mathbf{d}^\alpha}{d_0^\alpha} \right) \right]^{-1} \tag{4.12.10}$$

is the effective hydraulic permeability of the interstitial fluid solution (solvent and solutes) through the porous solid matrix; $\mathbf{k}$ is the hydraulic permeability of the solvent through the porous solid matrix; $\mathbf{d}^\alpha$ is the diffusivity of solute α through the mixture (frictional interactions with solvent and solid); and d_0^α is its free diffusivity (frictional interactions with solvent only). $\varphi^w \approx 1 - \varphi^s$ is the solid matrix porosity in the current configuration. The above expressions for the solvent and solute flux do not account for external body forces.

Also see Section 8.7 for additional guidelines for running multiphasic materials.

4.12.1 Guidelines for Multiphasic Analyses

4.12.1.1 Initial State of Swelling

Under traction-free conditions, a multiphasic material is usually in a state of swelling due to the osmotic pressure difference between the interstitial fluid and the external environment. This osmotic pressure arises from the difference in interstitial versus external concentrations of cations and anions, caused by the charged solid matrix and the requirement to satisfy electroneutrality. An osmotic pressure difference arising from such electrostatic interactions is known as the *Donnan osmotic pressure*. When the Donnan pressure is non-zero, traction-free conditions do not produce stress-free conditions for the solid matrix, since the matrix must expand until its stressed state resists the swelling pressure.

The Donnan pressure reduces to zero when the fixed charged density is zero, or when the external environment is infinitely hypertonic (having ion concentrations infinitely greater than the interstitial fixed charge density). Since these two conditions represent special cases, it is generally necessary to devise methods for achieving the desired initial state of swelling, in an analysis where loads or displacements need to be prescribed over and above this swollen state. Swelling occurs as a result of the influx of solvent into the porous solid matrix. This influx is a time-dependent process that could require extensive analysis time. Therefore, it is computationally efficacious to achieve the initial state of swelling by using a multi-step analysis (Chapter 6) where the first step is a steady-state analysis (Section 3.3.1). In this steady-state step, the fixed charge density may be ramped up from zero to the desired value using a load curve for that property or, alternatively, the external environmental conditions may be reduced from a very high hypertonic state down to the desired level. In the second step, prescribed displacement boundary conditions (when needed) may be specified to be of type *relative* (Section 3.11.1), so that they become superposed over and above the initial swelling state.

Example:

```
<Module type="multiphasic"/>
<Step>
  <Control>
    <analysis>STEADY-STATE</analysis>
    ...
  </Control>
</Step>
<Step>
  <Control>
    ...
  </Control>
  <Boundary>
    <bc type="prescribe" node_set="set1">
      <dof>z</dof>
      <scale>1</scale>
      <relative>1</relative>
    </prescribe>
  </Boundary>
</Step>
```

4.12.1.2 Prescribed Boundary Conditions

In most analyses, it may be assumed that the ambient fluid pressure and electric potential in the external environment are zero, thus $p_* = 0$ and $\psi_* = 0$, where the subscripted asterisk is used to denote environmental conditions. Since the external environment does not include a solid matrix, the fixed charge density there is zero. For example, in a triphasic analysis, $c_*^+ = c_*^- \equiv c_*$. It follows that the effective fluid pressure in the external environment is $\tilde{p}_* = -R\theta\Phi_* \sum_\alpha c_*^\alpha$ and the effective concentrations are $\tilde{c}_*^\alpha = c_*^\alpha/\hat{\kappa}_*^\alpha \tilde{c}_*^\alpha$. Therefore, in multiphasic analyses, whenever the external environment contains solutes with non-zero concentrations c_*^α, the user must remember to prescribe non-zero boundary conditions for the effective solute concentrations *and* the effective fluid pressure.

Letting $p_* = 0$ also implies that prescribed mixture normal tractions (Section 3.13.2.5) represent only the traction above ambient conditions. Note that users are not obligated to assume that $p_* = 0$. However, if a non-zero value is assumed for the ambient pressure, then users must remember to incorporate this non-zero value whenever prescribing mixture normal tractions. Similarly, users are not required to assume that $\psi_* = 0$; when a non-zero value is assumed for the electric potential of the external environment, the prescribed boundary conditions for the effective concentrations should be evaluated using the corresponding partition coefficient, $\tilde{c}_*^\alpha = c_*^\alpha/\tilde{\kappa}_*^\alpha \tilde{c}_*^\alpha$.

4.12.1.3 Prescribed Initial Conditions

When a multiphasic material is initially exposed to a given external environment with effective pressure $\tilde{p}_*$ and effective concentrations $\tilde{c}_*^\alpha$, the initial conditions inside the material should be set to $\tilde{p} = \tilde{p}_*$ and $\tilde{c}^\alpha = \tilde{c}_*^\alpha$ in order to expedite the evaluation of the initial state of swelling. The values of $\tilde{p}_*$ and $\tilde{c}_*^\alpha$ should be evaluated as described in Section 8.7.2.

4.12.1.4 Prescribed Effective Solute Flux

The finite element formulation for multiphasic materials in FEBio requires that the natural boundary condition for solute α be prescribed as $\tilde{j}_n^\alpha \equiv j_n^\alpha + \sum_\beta z^\beta j_n^\beta$, where $\tilde{j}_n^\alpha$ is the effective solute flux. For a mixture containing only neutral solutes ($z^\beta = 0, \forall\beta$), it follows that $\tilde{j}_n^\alpha = j_n^\alpha$.

4.12.1.5 Prescribed Electric Current Density

The electric current density in a mixture is a linear superposition of the ion fluxes,

$$\mathbf{I}_e = F_c \sum_\alpha z^\alpha \mathbf{j}^\alpha \,. \tag{4.12.11}$$

Since only the normal component $j_n^\alpha = \mathbf{j}^\alpha \cdot \mathbf{n}$ of ion fluxes may be prescribed at a boundary, it follows that only the normal component $I_n = \mathbf{I}_e \cdot \mathbf{n}$ of the current density may be prescribed. To prescribe I_n, it is necessary to know the nature of the ion species in the mixture, and how the current is being applied. For example, if the ions in the triphasic mixture consist of Na^+ and Cl^-, and if the current is being applied using silver/silver chloride (Ag/AgCl) electrodes, a chemical reaction occurs at the anode where Ag combines with Cl^- to produce AgCl, donating an electron to the electrode to transport the current. At the cathode, the reverse process takes place. Therefore, in this system, there is only flux of Cl^- and no flux of Na^+ ($j_n^+ = 0$) at the electrode-mixture interface, so that the prescribed boundary condition should be $j_n^- = -I_n/F_c$.

Since $z^+ = +1$ and $z^- = -1$ in a triphasic mixture, the corresponding effective fluxes are given by $\tilde{j}_n^+ = 2j_n^+ - j_n^- = I_n/F_c$ and $\tilde{j}_n^- = j_n^+ = 0$.

4.12.1.6 Electrical Grounding

If a multiphasic mixture containing ions is completely surrounded by ion-impermeant boundaries it is necessary to ground the mixture to prevent unconstrained fluctuations of the electric potential. Grounding is performed by prescribing the effective concentration of one or more ions, at a location inside the mixture or on one of its boundaries corresponding to the location of the grounding electrode. The choice of ion(s) to constrain may be guided by the type of grounding electrode and the reference electrolyte bath in which it is inserted.

4.12.1.7 Neglecting Osmotic Effects

Osmotic effects can be neglected in a multiphasic analysis by setting the osmotic coefficient to zero, $\Phi = 0$. In that case, the effective fluid pressure and the actual fluid pressure become equivalent measures, $\tilde{p} = p$, as for the case of a biphasic mixture.

4.12.1.8 Solutes as 'Solid-Bound Molecules'

In multiphasic analyses, FEBio solves for the solid displacement $\mathbf{u}$, the effective fluid pressure $\tilde{p}$, and the effective solute concentrations $\tilde{c}^\alpha$, at all the nodes, representing all the degrees of freedom in an analysis. Once the solution has converged at each time point, FEBio solves for the referential mass densities ρ_r^σ of solid-bound molecules by solving the corresponding mass balance equation in eq.(4.13.7) at integration points, in a single numerical integration step. This approach reduces the number of degrees of freedom that need to be solved simultaneously by the FEBio solver. However, occasionally, the numerical solution for ρ_r^σ may produce numerical round-off errors that may get amplified as the analysis time progresses.

An alternative to using solid-bound molecules is to define solutes σ whose diffusivity $\mathbf{d}^\sigma$ in the mixture is set to $\mathbf{0}$ (however, the free diffusivity d_0^σ should not be set to zero, to prevent a division by zero; its exact value is not important, thus let $d_0^\sigma = 1$). Then, FEBio treats these solutes as equivalent to solid-bound molecules: (1) Their concentration does not contribute to the osmolarity of the interstitial fluid; (2) their concentration contributes to the fixed charge density c^F in the current configuration; (3) these solutes do not contribute to the effective hydraulic permeability $\tilde{\mathbf{k}}$ of the porous multiphasic mixture; (4) these solutes can be involved in chemical reactions; (5) while their initial effective concentration may be prescribed, it must not contribute to the prescribed initial effective fluid pressure, and the user should not prescribe any boundary conditions on the effective concentration of these solutes. Using these 'solid-bound' solutes increases the number of degrees of freedom in an FEBio analysis; however, unlike solid-bound molecules, the solution for the effective concentration of these solutes remains as accurate as all other degrees of freedom in an analysis.

To evaluate the contribution of these solutes to the referential solid volume fraction φ_r^s and to chemical reactions, it should be recalled that the concentration c^σ is related to the referential mass density ρ_r^σ via eq.(4.13.8). Using the expression for φ_r^s in eq.(4.13.9), we find that

$$\varphi_r^s = \varphi_0^s + (J - \varphi_r^s) \sum_\sigma \frac{M^\sigma c^\sigma}{\rho_T^\sigma}$$

which can be solved for φ_r^s using the concentrations c^σ,

$$\varphi_r^s = \frac{\varphi_0^s + J \sum_\sigma \frac{M^\sigma c^\sigma}{\rho_T^\sigma}}{1 + \sum_\sigma \frac{M^\sigma c^\sigma}{\rho_T^\sigma}} .$$

4.12.2 General Specification of Multiphasic Materials

The material type for a multiphasic material is "*multiphasic*". Constitutive relations must be provided for the solid matrix, the mixture fixed charge density, the hydraulic permeability $\mathbf{k}$, the osmotic coefficient Φ, and the properties of each solute: the solute diffusivity in the mixture $\mathbf{d}^\alpha$, the solute free diffusivity d_0^α, and the solute effective solubility $\hat{\kappa}^\alpha$. Therefore, the following parameters must be defined:

<solid>	specification of the solid matrix	
<phi0>	solid volume fraction φ_r^s in the reference configuration	[ ]
<fixed_charge_density>	fixed charge density c_r^F in the reference configuration	[**n**/**L**^{3}]
<permeability>	specification of the hydraulic permeability $\mathbf{k}$	
<solvent_supply>	specification of the solvent supply $\hat{\varphi}^w$	
<osmotic_coefficient>	specification of the osmotic coefficient Φ	
<solute>	specification of the solute properties	
<solid_bound>	specification of solid-bound molecule	

The <solid> tag encloses a description of the solid matrix constitutive relation and associated material properties, as may be selected from the list provided in Section 4.1.4. The solid volume fraction in the reference configuration, <phi0>, must be greater than 0 (no solid) and less than 1 (only solid). The volume fraction of fluid (also known as the porosity) in the reference configuration is given by $1 - \varphi_0$. The <fixed_charge_density> may be negative, positive, or zero. The <permeability> and <osmotic_coefficient> tags enclose descriptions of the permeability and osmotic coefficient constitutive relations and their associated material properties, as may be selected from the list presented in Sections 4.10.2 and 4.11.5.

The optional <solute> tag provides a description of each solute in the multiphasic mixture. Multiple solutes may be defined. Each tag includes the required *sol* attribute, which should reference a solute *id* from the <Solutes> description in the <Globals> section (Section 3.4.2). The following parameters must be defined in this description:

<diffusivity>	specification of the solute diffusivities $\mathbf{d}^\alpha$ and d_0^α
<solubility>	specification of the solute effective solubility $\hat{\kappa}^\alpha$

The <diffusivity> and <solubility> tags enclose descriptions of materials that may be selected from the lists presented in Sections 4.11.3 and 4.11.4, respectively. Each solute tag must include a "sol" attribute

The optional <solid_bound> tag specifies which solid-bound molecule should be included in the multiphasic mixture. Multiple solid-bound molecules may be specified. Each tag should include the required *sbm* attribute, which references an *id* from the <SolidBoundMolecules> description in the <Globals> section (Section 3.4.3). The following parameter must be defined in this description:

<rho0>	initial value of the referential apparent density of the solid-bound molecule ρ_r^σ	[**M**/**L**^{3}]
<rhomin>	optional minimum allowable value of ρ_r^σ (zero by default)	[**M**/**L**^{3}]
<rhomax>	optional maximum allowable value of ρ_r^σ (none by default)	[**M**/**L**^{3}]

If a chemical reaction involves this solid-bound molecule its referential apparent density may evolve over time. The user may place lower and upper bounds on the allowable range of an evolving ρ_r^σ.

Example:

```
<material id="2" name="Media" type="multiphasic">
  <phi0>0.2</phi0>
  <fixed_charge_density>-40</fixed_charge_density>
  <solid name="Solid Matrix" type="Holmes-Mow">
    <density>1</density>
    <E>0.28</E>
    <v>0</v>
    <beta>0</beta>
  </solid>
  <permeability name="Permeability" type="perm-Holmes-Mow">
    <perm>1e-3</perm>
    <M>0</M>
    <alpha>0</alpha>
  </permeability>
  <osmotic_coefficient name="Ideal" type="osm-coef-const">
    <osmcoef>1.0</osmcoef>
  </osmotic_coefficient>
  <solute sol="1">
    <diffusivity name="Diffusivity" type="diff-const-iso">
      <free_diff>1e-3</free_diff>
      <diff>1e-3</diff>
    </diffusivity>
    <solubility name="Solubility" type="solub-const">
      <solub>1.0</solub>
    </solubility>
  </solute>
  <solute sol="2">
    <diffusivity name="Diffusivity" type="diff-const-iso">
      <free_diff>1e-3</free_diff>
      <diff>1e-3</diff>
    </diffusivity>
    <solubility name="Solubility" type="solub-const">
      <solub>1.0</solub>
    </solubility>
  </solute>
</material>
```

When a multiphasic material is employed in an analysis, it is also necessary to specify the values of the universal gas constant R [**F**·**L**/**n**·**T**], absolute temperature θ [**T**], and Faraday's constant F_c [**Q**/**n**] in the <Globals> section, using a self-consistent set of units.

Example:

```
<Globals>
```

```
    <Constants>
      <R>8.314e-6</R>
      <T>298</T>
      <Fc>96500e-9</Fc>
    </Constants>
    <Solutes>
      <solute id="1" name="Na">
        <charge_number>1</charge_number>
      </solute>
      <solute id="2" name="Cl">
        <charge_number>-1</charge_number>
      </solute>
    </Solutes>
  </Globals>
```

Example:

Self-consistent units for a triphasic analysis	
Primary Units	
time	s
length	mm
force	N
mole	nmol
charge	C
temperature	K
Derived Units	
stress	N/mm^2, MPa
permeability	mm^4/N·s, mm^2/MPa ·s
diffusivity	mm^2/s
concentration	nmol/mm^3, mM
charge density	nEq/mm^3, mEq/L
voltage	mV
current density	A/mm^2
current	A

It is also possible to create models with multiphasic materials that use different solutes in different regions. In that case, introduce additional solute entries in the <Solutes> section and refer to those solute ids in the multiphasic material descriptions. Generally, two adjoining multiphasic regions may share the same solute (e.g., Na in both regions), in which case that solute may transport freely across the interface separating these regions; or they may share no solute, in which case the interface is impermeable to all solutes.

4.12.3 Solvent Supply Materials

Solvent supply materials may be used to simulate an external source of solvent, such as supply from microvasculature that is not modeled explicitly. The solvent supply term, $\hat{\varphi}^w$, appears in the mass balance relation for the mixture,

$$\operatorname{div}\left(\mathbf{v}^s + \mathbf{w}\right) = \hat{\varphi}^w \,. \tag{4.12.12}$$

$\hat{\varphi}^w$ has units of reciprocal time [$\mathbf{t}^{-1}$]; it represents the rate at which the volume fraction of solvent changes with time.

4.12.3.1 Starling Equation

The material type for Starling's equation for fluid supply is *"Starling"*. The following material parameters need to be defined:

<kp>	hydraulic filtration coefficient, k_p	[**L**^{2}/**F**·**t**]
<pv>	effective fluid pressure in external source, $\tilde{p}_v$	[**P**]
<qc sol="n">	osmotic filtration coefficient, q_c^α	[**L**^{3}/**n**·**t**]
<cv sol="n">	effective solute concentration in external source, $\tilde{c}_v^\alpha$	[**n**/**L**^{3}]

The fluid supply is given by Starling's equation,

$$\hat{\varphi}^w = k_p \left(\tilde{p}_v - \tilde{p}\right) + \sum_\alpha q_c^\alpha \left(\tilde{c}_v^\alpha - \tilde{c}^\alpha\right) ,$$

where $\tilde{p}$ is the effective fluid pressure in the multiphasic material.

Example:

This example defines a solvent supply material of the Starling type for a multiphasic mixture containing one solute.

```
<solvent_supply type="Starling">
  <kp>0.001</kp>
  <pv>0.1</pv>
  <qc sol="1">1e-5</qc>
  <cv sol="1">150</cv>
</solvent_supply>
```

4.13 Chemical Reactions

The following sections describe FEBio's formulation for analyzing mechanochemical events in neutral or charged deformable porous media under finite deformation. The formulation allows full coupling of mechanical and chemical effects, providing a framework where material properties and response functions may depend on solid matrix strain as well as solute concentration.

If you use these capabilities in your published research, we request that you cite the following publication describing its development [9].

4.13.1 Guidelines for Chemical Reaction Analyses

Chemical reactions may be modeled within a multiphasic mixture. The reaction may involve solutes ($\alpha = \iota$) and solid-bound molecules ($\alpha = \sigma$) that move with the solid matrix ($\mathbf{v}^\sigma = \mathbf{v}^s, \forall\sigma$). Consider a general chemical reaction,

$$\sum_\alpha \nu_R^\alpha \mathcal{E}^\alpha \rightarrow \sum_\alpha \nu_P^\alpha \mathcal{E}^\alpha, \tag{4.13.1}$$

where $\mathcal{E}^\alpha$ is the chemical species representing constituent α in the mixture; ν_R^α and ν_P^α represent stoichiometric coefficients of the reactants and products, respectively. To maintain consistency with classical chemical kinetics, the analysis of chemical reactions employs molar concentrations c^α and molar supplies $\hat{c}^\alpha$ on a solution-volume basis for all reactants and products, whether they are solutes or solid-bound molecular species.

Since the molar supply of reactants and products is constrained by stoichiometry, it follows that all molar supplies $\hat{c}^\alpha$ in a specific chemical reaction may be related to a *molar production rate* $\hat{\zeta}$ according to

$$\hat{c}^\alpha = \nu^\alpha \hat{\zeta}\,, \tag{4.13.2}$$

where ν^α represents the net stoichiometric coefficient for $\mathcal{E}^\alpha$,

$$\nu^\alpha = \nu_P^\alpha - \nu_R^\alpha. \tag{4.13.3}$$

Thus, formulating constitutive relations for $\hat{c}^\alpha$ is equivalent to providing a single relation for $\hat{\zeta}\,(\theta, \mathbf{F}, c^\alpha)$. When the chemical reaction is reversible,

$$\sum_\alpha \nu_R^\alpha \mathcal{E}^\alpha \rightleftharpoons \sum_\alpha \nu_P^\alpha \mathcal{E}^\alpha, \tag{4.13.4}$$

the relations of (4.13.2)-(4.13.3) still apply but the constitutive relation for $\hat{\zeta}$ would be different.

Example:

Consider the dissociation of $CaCl_2$ into ions Ca^{2+} and Cl^-,

$$CaCl_2 \rightleftharpoons Ca^{2+} + 2Cl^-.$$

The mixture contains three constituents. The stoichiometric coefficients of the reactants are $\nu_R^{CaCl_2} = 1$, $\nu_R^{Ca^{2+}} = 0$, $\nu_R^{Cl^-} = 0$, and those of the products are $\nu_P^{CaCl_2} = 0$, $\nu_P^{Ca^{2+}} = 1$, $\nu_P^{Cl^-} = 2$.

The reaction production rate $\hat{\zeta}$ enters into the governing equations of multiphasic mixtures via the mass balance relation for each solute,

$$\frac{1}{J}\frac{D^s\left[J\left(1-\varphi^s\right)c^\iota\right]}{Dt} + \operatorname{div}\mathbf{j}^\iota = \left(1-\varphi^s\right)\nu^\iota\hat{\zeta}\,, \tag{4.13.5}$$

the mass balance for the mixture,

$$\operatorname{div}\left(\mathbf{v}^s + \mathbf{w}\right) = \left(1 - \varphi^s\right)\hat{\zeta}\overline{\mathcal{V}}\,, \tag{4.13.6}$$

where $\overline{\mathcal{V}} = \sum_\alpha \nu^\alpha \mathcal{V}^\alpha$ and $\mathcal{V}^\alpha = M^\alpha/\rho_T^\alpha$ is the molar volume of α, and the mass balance for solid-bound constituents,

$$D^s\rho_r^\sigma/Dt = \hat{\rho}_r^\sigma\,, \tag{4.13.7}$$

where ρ_r^σ is the referential apparent mass density (mass of σ per mixture volume in the reference configuration), and $\hat{\rho}_r^\sigma$ is the referential apparent mass supply of solid constituent σ, related to molar concentrations and supplies via

$$c^\sigma = \frac{\rho_r^\sigma}{\left(J - \varphi_r^s\right)M^\sigma}, \quad \hat{c}^\sigma = \frac{\hat{\rho}_r^\sigma}{\left(J - \varphi_r^s\right)M^\sigma}\,. \tag{4.13.8}$$

Internally, the content of solid-bound species is stored in ρ_r^σ and (4.13.8) is used to evaluate c^σ when needed for the calculation of $\hat{\zeta}$. If a solid-bound molecule is involved in a chemical reaction, equation (4.13.7) is integrated to produce an updated value of ρ_r^σ, using $\hat{\rho}_r^\sigma = \left(J - \varphi_r^s\right)M^\sigma\nu^\sigma\hat{\zeta}$ based on (4.13.2) and (4.13.8).

Evolving solid content due to chemical reactions implies that the referential solid volume fraction φ_r^s may not remain constant. This value is updated at every time point using

$$\varphi_r^s = \varphi_0^s + \sum_\sigma \frac{\rho_r^\sigma}{\rho_T^\sigma}\,, \tag{4.13.9}$$

where φ_0^s is the solid volume fraction specified by the multiphasic material parameter *phi0* (Section 4.12.2). Thus, φ_0^s may be used to account for the solid volume fraction not contributed explicitly by solid-bound molecules. Based on kinematics, the solid volume fraction in the current configuration is given by $\varphi^s = \varphi_r^s/J$. Therefore, since $0 \leqslant \varphi^s \leqslant 1$ by definition, it follows that $0 \leqslant \varphi_r^s \leqslant J$, implying that the referential solid volume fraction may evolve to values greater than unity when growth leads to swelling of the multiphasic mixture.

Similarly, if solid-bound molecules are charged and their content evolves over time, the referential fixed charge density may also evolve with chemical reactions according to

$$c_r^F = c_0^F + \frac{1}{1 - \varphi_r^s}\sum_\sigma \frac{z^\sigma\rho_r^\sigma}{M^\sigma}\,, \tag{4.13.10}$$

where c_0^F is the referential fixed charge density specified by the multiphasic material parameter *fixed_charge_density* (Section 4.12.2). Thus, c_0^F may be used to account for the fixed charge density not contributed explicitly by solid-bound molecules.

A chemical reaction is properly balanced when

$$\sum_\alpha \nu^\alpha M^\alpha = 0\,, \tag{4.13.11}$$

where M^α is the molar mass of α. This constraint implies that the net gain in mass of products must be the same as the net loss in mass of reactants. However, this constraint is not verified in the code, allowing users to model chemical reactions with implicit constituents (constituents that are neither explicitly modeled as solutes nor as solid-bound molecules, for which ν^α and M^α are

not given). For example, a chemical reaction where cells consume glucose to form a protein from amino-acids building blocks may have the form

$$\text{cells} + \text{glucose} + \text{amino acids} \rightarrow \text{cells} + \text{protein} + \text{waste products}\,.$$

The user may opt to model only the glucose reactant and the protein product explicitly, while the presence of all other species in this reaction is implicit. In these types of analyses the user must beware of potential inconsistencies in the evolving mass of reactants and products since only some of those constituents are modeled explicitly. In particular, the evolution of φ_r^s as given in (4.13.9) can only account for the explicitly modeled solid-bound molecules. Furthermore, when some reactants and products are implicit, the value of the reaction molar volume $\overline{\mathcal{V}}$ calculated in the code becomes inaccurate and may produce unexpected results in the evaluation of the mixture mass balance relation in (4.13.6). Therefore, the user is given the option to override the value of $\overline{\mathcal{V}}$ calculated in the code. In particular, if the precise molar volumes of all the species in a reaction are not known, assuming that $\overline{\mathcal{V}} \approx 0$ is a reasonable choice equivalent to assuming that all the constituents have approximately the same density ρ_T^α, as may be deduced from (4.13.11).

Since the electroneutrality condition is enforced in multiphasic mixtures in FEBio, it follows that chemical reactions must not violate this condition. Enforcing electroneutrality in a chemical reaction is equivalent to satisfying

$$\sum_\alpha z^\alpha \nu^\alpha = 0\,. \tag{4.13.12}$$

This constraint is checked within the code and an error is generated when it is violated.

A constitutive relation must be provided for the molar production rate $\hat{\zeta}\left(\theta, \mathbf{F}, c^\alpha\right)$ of each chemical reaction.

4.13.2 General Specification for Chemical Reactions

The material type for a chemical reaction is "*reaction*". The <reaction> tag must appear inside the definition of a multiphasic mixture and the reaction is defined only for that mixture. Multiple chemical reactions may be defined in a given mixture. Each <reaction> tag must include a *type* attribute that identifies the constitutive relation for $\hat{\zeta}$.

The stoichiometric coefficients ν_R^α of the reactants and ν_P^α for the products must be specified in every reaction. Optionally, the net reaction molar volume $\overline{\mathcal{V}}$ may be specified explicitly to override the internal value calculated in the code. Therefore, the following parameters need to be defined in a chemical reaction:

<vR>	reactant stoichiometric coefficient ν_R^α	[ ]
<vP>	product stoichiometric coefficient ν_P^α	[ ]
<Vbar>	optional override value for $\overline{\mathcal{V}}$	[$\mathbf{L}^3$/**n**]

Each <vR> and <vP> tag must include either the *sol* attribute, which should reference a solute *id* from the <Solutes> description in the <Globals> section (Section 3.4.2), or the *sbm* attribute, which should reference a solid-bound molecule *id* from the <SolidBoundMolecules> description in the <Globals> section (Section 3.4.3). Only solutes and solid-bound molecules that have been included in the parent multiphasic mixture may be specified as reactants or products of a chemical reaction.

Additional parameters may be needed in the definition of a chemical reaction, depending on the specific form of the constitutive relation for the production rate.

4.13.3 Chemical Reaction Materials

4.13.3.1 Law of Mass Action for Forward Reactions

The material type for the Law of Mass Action for a forward reaction is *mass-action-forward*. The following parameters must be defined:

<forward_rate>	specific forward reaction rate k

For this type of reaction the constitutive relation for the molar production rate is given by

$$\hat{\zeta} = k \prod_{\alpha} (c^{\alpha})^{\nu_R^{\alpha}} \ .$$

The constitutive form of the specific forward reaction rate must be selected from the list of materials given in Section 4.13.4. The units of $\hat{\zeta}$ are [**n**/**L**^{3}·**t**] and those of c^{α} are [**n**/**L**^{3}].

Example:

Consider the forward reaction that produces solid copper sulfide from solid copper and solid sulfur,

$$\mathrm{Cu} + \mathrm{S} \rightarrow \mathrm{CuS}$$

All three species are modeled explicitly in the mixture as solid-bound molecules, with id's 1 (Cu), 2 (for S) and 3 (for CuS). The chemical reaction material is given by:

```
<reaction name="copper sulfide production" type="mass-action-forward">
  <vR sbm="1">1</vR>
  <vR sbm="2">1</vR>
  <vP sbm="3">1</vP>
  <forward_rate type="constant reaction rate">
    <k>1e-3</k>
  </forward_rate>
</reaction>
```

4.13.3.2 Law of Mass Action for Reversible Reactions

The material type for the Law of Mass Action for a reversible reaction is *mass-action-reversible*. The following parameters must be defined:

<forward_rate>	specific forward reaction rate k_F
<reverse_rate>	specific reverse reaction rate k_R

For this type of reaction the constitutive relation for the molar production rate is given by

$$\hat{\zeta} = \hat{\zeta}_F - \hat{\zeta}_R \,,$$

where

$$\hat{\zeta}_F = k_F \prod_\alpha (c^\alpha)^{\nu_R^\alpha}$$
$$\hat{\zeta}_R = k_R \prod_\alpha (c^\alpha)^{\nu_P^\alpha} \,.$$

The constitutive form of the specific forward and reverse reaction rates must be selected from the list of materials given in Section 4.13.4. The units of $\hat{\zeta}_F$ and $\hat{\zeta}_R$ are [**n**/**L**^{3}·**t**] and those of c^α are [**n**/**L**^{3}].

Example:

Consider the reversible dissociation of CaCl salt into Ca^{2+} and Cl^- in water,

$$CaCl_2 \rightleftharpoons Ca^{2+} + 2Cl^-$$

All three species are modeled explicitly in the mixture as solutes, with id's 1 (for $CaCl_2$), 2 (for Ca^{2+}) and 3 (for Cl^-). The chemical reaction material is given by:

```
<reaction name="CaCl2 dissociation" type="mass-action-reversible">
  <vR sol="1">1</vR>
  <vP sol="2">1</vP>
  <vP sol="3">2</vP>
  <forward_rate type="constant">
    <k>1.0</k>
  </forward_rate>
  <reverse_rate type="constant">
    <k>0.1</k>
  </reverse_rate>
</reaction>
```

4.13.3.3 Michaelis-Menten Reaction

The material type for the Michaelis-Menten reaction is *Michaelis-Menten*. The following parameters must be defined:

<forward_rate>	maximum rate at saturating substrate concentration $V_{\max}$	[**n**/**L**^{3}·**t**]
<Km>	substrate concentration when reaction rate is half of $V_{\max}$	[**n**/**L**^{3}]
<c0>	optional minimum substrate concentration to trigger reaction	[**n**/**L**^{3}]

The Michaelis-Menten reaction may be used to model enzyme kinetics where the enzyme $\mathcal{E}^e$ triggers the conversion of the substrate $\mathcal{E}^s$ into the product $\mathcal{E}^p$. The product molar supply is given by

$$\hat{c}^p = \begin{cases} \frac{V_{\max} c^s}{K_m + c^s} & c^s \geqslant c_0 \\ 0 & c^s < c_0 \end{cases},$$

where c^s is the substrate concentration. The default value of c_0 is 0. This relation may be derived, with some simplifying assumptions, by applying the law of mass action to the combination of two reactions,

$$\mathcal{E}^e + \mathcal{E}^s \rightleftharpoons \mathcal{E}^{es} \rightarrow \mathcal{E}^e + \mathcal{E}^p$$

Since the enzyme is not modeled explicitly in the Michaelis-Menten approximation to these two reactions, this reaction model is effectively equivalent to

$$\mathcal{E}^s \rightarrow \mathcal{E}^p.$$

Therefore, this reaction accepts only one reactant tag <vR> and one product tag <vP>. For consistency with the formulation of this reaction, the stoichiometric coefficients should be set to $\nu_R^s = \nu_P^p = 1$, so that $\hat{c}^p = \hat{\zeta}$.

The constitutive form of the specific forward reaction rate $V_{\max}$ must be selected from the list of materials given in Section 4.13.4.

Example:

```
<reaction name="enzyme kinetics" type="Michaelis-Menten">
  <Vbar>0</Vbar>
  <vR sol="1">1</vR>
  <vP sol="2">1</vP>
  <forward_rate type="constant">
    <k>1.0</k>
  </forward_rate>
  <Km>5.0</Km>
</reaction>
```

4.13.4 Specific Reaction Rate Materials

The following sections define specific materials that can be used to define the reaction rate of a chemical reaction.

4.13.4.1 Constant Reaction Rate

The material type for a constant specific reaction rate is *constant reaction rate*. The following parameter must be defined:

<k>	constant specific reaction rate k	[*units vary*]

Example:

```
<reaction name="enzyme kinetics" type="Michaelis-Menten">
  <Vbar>0</Vbar>
  <vR sol="1">1</vR>
  <vP sol="2">1</vP>
  <forward_rate type="constant reaction rate">
    <k>1.0</k>
  </forward_rate>
  <Km>5.0</Km>
</reaction>
```

4.13.4.2 Huiskes Reaction Rate

The material type for the Huiskes reaction rate is *Huiskes reaction rate*. The following parameters must be defined:

<B>	reaction rate per specific strain energy B	[*units vary*]
<psi0>	specific strain energy at homeostasis ψ_0	[**F**·**L**/**M**]

The Huiskes specific reaction rate has the form

$$k = \frac{B}{J - \varphi_r^s} \left(\frac{\Psi_r}{\rho_r^s} - \psi_0 \right) ,$$

where Ψ_r is the strain energy density of the solid (strain energy in current configuration per mixture volume in the reference configuration) and $\rho_r^s = \sum_\sigma \rho_r^\sigma$ is the referential apparent solid density (mass of solid in current configuration per mixture volume in reference configuration). The ratio Ψ_r/ρ_r^s is the specific strain energy (strain energy per mass of solid in the current configuration). The Huiskes specific reaction rate may assume positive and negative values; it reduces to zero at homeostasis, when $\Psi_r/\rho_r^s = \psi_0$.

Example:

To reproduce the interstitial solid remodeling theory proposed by Weinans et al. [81], consider the forward reaction

$$\text{cells} + \text{solutes} \rightarrow \text{cells} + \text{solid} ,$$

where cells convert solutes (e.g., nutrients) into synthesized solid when the specific strain energy exceeds the homeostatic value. If the cells and solutes are implicit in this reaction (e.g., if it is assumed that their concentrations change negligibly), the production rate of the solid may be given by $\hat{\zeta} = k$ using the law of mass action for a forward rection, where k has the form given above.

```
<reaction name="solid remodeling" type="mass-action-forward">
  <vP sbm="1">1</vP>
  <forward_rate type="Huiskes reaction rate">
    <B>1.0</B>
    <psi0>0.01</psi0>
  </forward_rate>
  <Km>5.0</Km>
</reaction>
```

4.14 Rigid Body

A rigid body can be defined using the rigid material model. The material type for a rigid body is "*rigid body*". The following parameters are defined:

<density>	Density of rigid body	[**M**/**L**^{3}]
<center_of_mass>	Position of the center of mass	[**L**]
<E>	Young's modulus	[**P**]
<v>	Poisson's ratio	[ ]

If the *center_of_mass* parameter is omitted, FEBio will calculate the center of mass automatically. In this case, a density *must* be specified. If the *center_of_mass* is defined the *density* parameter may be omitted. Omitting both will generate an error message.

The Young's modulus $E$and Poisson ratio v currently have no effect on the results. The only place where they are used is in the calculation of a material stiffness for some auto-penalty contact formulation. If you are using contact it is advised to enter sensible values. Otherwise these parameters may be omitted.

The degrees of freedom of a rigid body are initially unconstrained[3]. This implies that a rigid body is free to move in all three directions and rotate about any of the coordinate axes. To constrain a rigid body degree of freedom you may use the *Constraints* sections (Section 3.12.1).

Example:

```
<material id="1" type="rigid body">
  <density>1.0</density>
  <center_of_mass>0,0,0</center_of_mass>
</material>
```

[3]This is different from previous versions of FEBio where rigid bodies were initially fully constrained.

4.15 Active Contraction

Active contraction materials may be used to impose a (typically) contractile stress within a solid material, by incorporating the contraction material within a solid mixture. The total stress in the solid mixture is simply $\boldsymbol{\sigma} = \boldsymbol{\sigma}^s + \boldsymbol{\sigma}^a$, where $\boldsymbol{\sigma}^s$ is the solid stress (due to strain and strain history), and $\boldsymbol{\sigma}^a$ is the active contractile stress. The calculation of the contractile stress is the same, wether the solid mixture consists of uncoupled or unconstrained materials.

4.15.1 Contraction in Mixtures of Uncoupled Materials

When the solid mixture consists of uncoupled materials, the active contraction material should be selected from the list below.

4.15.1.1 Uncoupled Prescribed Uniaxial Active Contraction

The material type for prescribed active contraction along a single direction (or fiber) in an uncoupled solid mixture is "*uncoupled prescribed uniaxial active contraction*". This material must be combined with a stable uncoupled material that acts as a passive matrix, using a "*uncoupled solid mixture*" container as described in Section 4.1.2.16. The following material parameters need to be defined:

<T0>	T_0, representing the prescribed stress	[**P**]

In the reference configuration, the fiber is oriented along the unit vector $\mathbf{e}_1$, where $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ are orthonormal basis vectors representing the local element coordinate system when specified (Section 4.1.1), or else the global Cartesian coordinate system. The active stress $\boldsymbol{\sigma}^a$ for this material is given by

$$\boldsymbol{\sigma}^a = J^{-1} T_0 \mathbf{n} \otimes \mathbf{n} \,,$$

where $\mathbf{n} = \mathbf{F} \cdot \mathbf{n}_r$ is the stretched fiber orientation in the current (deformed) configuration.

Example:

Uniaxial contraction along $\mathbf{e}_1$, in a mixture containing a Mooney-Rivlin solid.

```
<material id="1" type="uncoupled solid mixture">
  <mat_axis type="local">0,0,0</mat_axis>
  <k>1000</k>
  <solid type="Mooney-Rivlin">
    <c1>1.0</c1>
    <c2>0</c2>
  </solid>
  <solid type="prescribed uniaxial active contraction uncoupled">
    <T0 lc="2">1</T0>
    <mat_axis type="angles">
      <theta>0</theta>
      <phi>90</phi>
    </mat_axis>
  </solid>
</material>
```

4.15.1.2 Uncoupled Prescribed Transversely Isotropic Active Contraction

The material type for prescribed isotropic active contraction in a plane transverse to a given direction (or fiber), in an uncoupled solid mixture, is "*prescribed trans iso active contraction uncoupled*". This material must be combined with a stable uncoupled material that acts as a passive matrix, using a "*uncoupled solid mixture*" container as described in Section 4.1.2.16. The following material parameters need to be defined:

<T0>	T_0, representing the prescribed stress	[**P**]

In the reference configuration, the fiber is oriented along the unit vector $\mathbf{e}_1$, where $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ are orthonormal basis vectors representing the local element coordinate system when specified

(Section 4.1.1), or else the global Cartesian coordinate system. The active stress $\boldsymbol{\sigma}^a$ for this material is given by

$$\boldsymbol{\sigma}^a = J^{-1} T_0 \left(\mathbf{B} - \mathbf{n} \otimes \mathbf{n} \right) ,$$

where $\mathbf{n} = \mathbf{F} \cdot \mathbf{e}_1$ is the stretched fiber orientation in the current (deformed) configuration and $\mathbf{B} = \mathbf{F} \cdot \mathbf{F}^T$ is the left Cauchy-Green tensor.

Example:
Isotropic contraction in plane transverse to $\mathbf{e}_1$, in a mixture containing a Mooney-Rivlin solid.

```
<material id="1" type="uncoupled solid mixture">
  <mat_axis type="local">0,0,0</mat_axis>
  <solid type="Mooney-Rivlin">
    <c1>1.0</c1>
    <c2>0</c2>
    <k>1000</k>
  </solid>
  <solid type="prescribed trans iso active contraction uncoupled">
    <T0 lc="2">1</T0>
    <mat_axis type="angles">
      <theta>0</theta>
      <phi>90</phi>
    </mat_axis>
  </solid>
</material>
```

4.15.1.3 Uncoupled Prescribed Isotropic Active Contraction

The material type for prescribed isotropic active contraction, in an uncoupled solid mixture, is "*prescribed isotropic active contraction uncoupled*". This material must be combined with a stable uncoupled material that acts as a passive matrix, using a "*uncoupled solid mixture*" container as described in Section 4.1.2.16. The following material parameters need to be defined:

<T0>	T_0, representing the prescribed stress	[**P**]

The active stress $\boldsymbol{\sigma}^a$ for this material is given by

$$\boldsymbol{\sigma}^a = J^{-1} T_0 \mathbf{B} \, .$$

Note: If the solid material in the mixture is (nearly) incompressible, this isotropic contraction will cause no change in the deformation.

Example:
Isotropic contraction in a mixture containing a Mooney-Rivlin solid.

```
<material id="1" type="uncoupled solid mixture">
  <mat_axis type="local">0,0,0</mat_axis>
  <solid type="Mooney-Rivlin">
    <c1>1.0</c1>
    <c2>0</c2>
    <k>5.0</k>
```

```
  </solid>
  <solid type="prescribed isotropic active contraction uncoupled">
    <T0 lc="2">1</T0>
  </solid>
</material>
```

4.15.1.4 Prescribed Fiber Stress

An active fiber stress, based on a Hill formulation, can be added via the material "*uncoupled active fiber stress*". This material must be combined with a stable compressible material that acts as a passive matrix, using a "*uncoupled solid mixture*" container as described in Section 4.1.2.16. The stress is given by,

$$\boldsymbol{\sigma}^a = J^{-1} T\left(\tilde{\lambda}\right) \mathrm{dev}\mathbf{A}.$$

Here, $\mathbf{A} = \mathbf{a} \otimes \mathbf{a}$,with **a** the unit vector describing the fiber direction in the spatial frame, $\tilde{\lambda}$is the deviatoric fiber stretch, *J* is the jacobian of the deformation, and

$$T\left(\tilde{\lambda}\right) = s_{max} a\left(t\right) s_{TL}\left(\tilde{\lambda}\right) s_{TV}\left(\dot{\tilde{\lambda}}\right)$$

The parameters are defined below.

smax	scale factor for defining maximum stress	[**P**]
a	activation level	
stl	Function defining the stress-stretch relatioship for the material	
stv	Function defining the stress-stretch rate relatioship for the material.	

The parameters$s_{TL}$and $s_{TV}$are functions that need to be defined in place. There are currently two ways of defining these functions, either via a mathematical expression or a list of sample points. An example is given below. If these parametes are omitted, they are replaced by the constant 1 in the equation for the stress above.

Example:

An example defining the stl parameter via a mathematical expression.

```
<material id="1" type="uncoupled solid mixture">
  <k>100.0</k>
  <mat_axis type="local">0,0,0</mat_axis>
  <solid type="Mooney-Rivlin">
    <c1>1.0</c1>
    <c2>0</c2>
  </solid>
  <solid type="uncoupled active fiber stress">
    <smax>1.5</smax>
    <a lc="1">0.1</a>
    <stl type="math">
      <math>(l-1)^2</math>
    </stl>
  </solid>
</material>
```

An example defining the stl parameter via a point list.

```
<material id="1" type="uncoupled solid mixture">
  <k>100.0</k>
  <mat_axis type="local">0,0,0</mat_axis>
  <solid type="Mooney-Rivlin">
    <c1>1.0</c1>
    <c2>0</c2>
  </solid>
  <solid type="uncoupled active fiber stress">
    <smax>1.5</smax>
    <a lc="1">0.1</a>
    <stl type="point">
      <interpolate>linear</interpolate>
      <points>
        <pt>0,0</pt>
        <pt>1,0</pt>
        <pt>2,1</pt>
      </points>
    </stl>
  </solid>
</material>
```

4.15.2 Contraction in Mixtures of Unconstrained Materials

When the solid mixture consists of unconstrained materials, the active contraction material should be selected from the list below.

4.15.2.1 Prescribed Uniaxial Active Contraction

The material type for prescribed active contraction along a single direction (or fiber) in a solid mixture of unconstrained materials is "*prescribed uniaxial active contraction*". This material must be combined with a stable compressible material that acts as a passive matrix, using a "*solid mixture*" container as described in Section 4.1.4.26. The following material parameters need to be defined:

<T0>	T_0, representing the prescribed stress	[**P**]

In the reference configuration, the fiber is oriented along the unit vector $\mathbf{e}_1$, where $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ are orthonormal basis vectors representing the local element coordinate system when specified (Section 4.1.1), or else the global Cartesian coordinate system. The active stress $\boldsymbol{\sigma}^a$ for this material is given by

$$\boldsymbol{\sigma}^a = J^{-1} T_0 \mathbf{n} \otimes \mathbf{n} \,,$$

where $\mathbf{n} = \mathbf{F} \cdot \mathbf{e}_1$ is the stretched fiber orientation in the current (deformed) configuration.

Example:

Uniaxial contraction along $\mathbf{e}_1$, in a mixture containing a neo-Hookean solid.

```
<material id="1" type="solid mixture">
  <mat_axis type="local">0,0,0</mat_axis>
  <solid type="neo-Hookean">
    <E>1.0</E>
    <v>0.3</v>
  </solid>
  <solid type="prescribed uniaxial active contraction">
    <T0 lc="2">1</T0>
    <mat_axis type="angles">
      <theta>0</theta>
      <phi>90</phi>
    </mat_axis>
  </solid>
</material>
```

4.15.2.2 Prescribed Transversely Isotropic Active Contraction

The material type for prescribed isotropic active contraction in a plane transverse to a given direction (or fiber), in a solid mixture of unconstrained materials, is "*prescribed trans iso active contraction*". This material must be combined with a stable compressible material that acts as a passive matrix, using a "*solid mixture*" container as described in Section 4.1.4.26. The following material parameters need to be defined:

<T0>	T_0, representing the prescribed stress	[**P**]

In the reference configuration, the fiber is oriented along the unit vector $\mathbf{e}_1$, where $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ are orthonormal basis vectors representing the local element coordinate system when specified (Section 4.1.1), or else the global Cartesian coordinate system. The active stress $\boldsymbol{\sigma}^a$ for this material is given by

$$\boldsymbol{\sigma}^a = J^{-1} T_0 \left(\mathbf{B} - \mathbf{n} \otimes \mathbf{n} \right) ,$$

where $\mathbf{n} = \mathbf{F} \cdot \mathbf{e}_1$ is the stretched fiber orientation in the current (deformed) configuration and $\mathbf{B} = \mathbf{F} \cdot \mathbf{F}^T$ is the left Cauchy-Green tensor.

Example:

Isotropic contraction in plane transverse to $\mathbf{e}_1$, in a mixture containing a neo-Hookean solid.

```
<material id="1" type="solid mixture">
  <mat_axis type="local">0,0,0</mat_axis>
  <solid type="neo-Hookean">
    <E>1.0</E>
    <v>0.3</v>
  </solid>
  <solid type="prescribed trans iso active contraction">
    <T0 lc="2">1</T0>
    <mat_axis type="angles">
      <theta>0</theta>
      <phi>90</phi>
    </mat_axis>
  </solid>
</material>
```

4.15.2.3 Prescribed Isotropic Active Contraction

The material type for prescribed isotropic active contraction, in a solid mixture of unconstrained materials, is "*prescribed isotropic active contraction*". This material must be combined with a stable compressible material that acts as a passive matrix, using a "*solid mixture*" container as described in Section 4.1.4.26. The following material parameters need to be defined:

<T0>	T_0, representing the prescribed stress	[**P**]

The active stress $\boldsymbol{\sigma}^a$ for this material is given by

$$\boldsymbol{\sigma}^a = J^{-1} T_0 \mathbf{B} .$$

Example:

Isotropic contraction in a mixture containing a neo-Hookean solid.

```
<material id="1" type="solid mixture">
  <mat_axis type="local">0,0,0</mat_axis>
  <solid type="neo-Hookean">
    <E>1.0</E>
    <v>0</v>
  </solid>
  <solid type="prescribed isotropic active contraction">
    <T0 lc="2">1</T0>
  </solid>
</material>
```

4.15.2.4 Prescribed Fiber Stress

An active fiber stress, based on a Hill formulation, can be added via the material "*active fiber stress*". This material must be combined with a stable compressible material that acts as a passive matrix, using a "*solid mixture*" container as described in Section 4.1.4.26. The stress is given by,

$$\boldsymbol{\sigma}^a = J^{-1} T(\lambda)\, \mathbf{A}.$$

Here, $\mathbf{A} = \mathbf{a} \otimes \mathbf{a}$,with **a** the unit vector describing the fiber direction in the spatial frame, $\lambda$is the fiber stretch, *J* is the jacobian of the deformation, and

$$T(\lambda) = s_{max} a(t)\, s_{TL}(\lambda)\, s_{TV}\left(\dot{\lambda}\right)$$

The parameters are defined below.

smax	scale factor for defining maximum stress	[**P**]
a	activation level	
stl	Function defining the stress-stretch relatioship for the material	
stv	Function defining the stress-stretch rate relatioship for the material.	

The parameters$s_{TL}$and $s_{TV}$are functions that need to be defined in place. There are currently two ways of defining these functions, either via a mathematical expression or a list of sample points. An example is given below. If these parametes are omitted, they are replaced by the constant 1 in the equation for the stress above.

Example:

An example defining the stl parameter via a mathematical expression.

```
<material id="1" type="solid mixture">
  <mat_axis type="local">0,0,0</mat_axis>
  <solid type="neo-Hookean">
    <E>1.0</E>
    <v>0</v>
  </solid>
  <solid type="active fiber stress">
    <smax>150</smax>
    <a lc="1">0.1</a>
    <stl type="math">
      <math>(l-1)^2</math>
    </stl>
  </solid>
</material>
```

An example defining the stl parameter via a point list.

```
<material id="1" type="solid mixture">
  <mat_axis type="local">0,0,0</mat_axis>
  <solid type="neo-Hookean">
    <E>1.0</E>
    <v>0</v>
```

```
    </solid>
    <solid type="active fiber stress">
      <smax>150</smax>
      <a lc="1">0.1</a>
      <stl type="point">
        <interpolate>linear</interpolate>
        <points>
          <pt>0,0</pt>
          <pt>1,0</pt>
          <pt>2,1</pt>
        </points>
      </stl>
    </solid>
  </material>
```

4.16 Viscous Fluids

Fluid mechanics analyses may be used to examine fluid flow over fixed domains. Fluid-structure interactions (FSI) may be used to examine fluid flow over deforming domains. It is also possible to perform solute (mass) transport analyses in fluid mixtures using the fluid-solutes module. FEBio's fluid, fluid-FSI and fluid-solutes modules treat the fluid as isothermal (constant and uniform temperature) and compressible; incompressible flow is simulated by prescribing a physically realistic bulk modulus K to model the fluid's elastic compressibility (for example, $K = 2.2\,\text{GPa}$ for water at room temperature). The Cauchy stress $\boldsymbol{\sigma}^f$ in a fluid is given by

$$\boldsymbol{\sigma}^f = -p\mathbf{I} + \boldsymbol{\tau}\,, \tag{4.16.1}$$

where p is the fluid pressure arising from the elastic and osmotic responses and $\boldsymbol{\tau}$ is the viscous stress resulting from the fluid viscosity and its rate of deformation. FEBio's fluid solver uses the fluid dilatation e^f, instead of the *effective fluid pressure* $\tilde{p}$, as a nodal degree of freedom. The effective fluid pressure $\tilde{p}$ may differ from the actual fluid pressure p if solutes are present in the fluid mixture, as per eq.(4.12.7). In fluid analyses the effective fluid pressure $\tilde{p}$ is also the elastic contribution to the fluid pressure, resulting from changes in e^f. FEBio provides several alternative constitutive models for the elastic response $\tilde{p}\left(e^f\right)$, including

$$\begin{aligned}
\tilde{p}\left(e^f\right) &= -Ke^f && \text{linear}\\
\tilde{p}\left(e^f\right) &= K\left(\frac{1}{1+e^f}-1\right) && \text{nonlinear}\\
\tilde{p}\left(e^f\right) &= -K\frac{\ln\left(1+e^f\right)}{1+e^f} && \text{log nonlinear}
\end{aligned} \tag{4.16.2}$$

where K is the fluid's bulk modulus (a physical property that may be looked up or measured). The reason for selecting e^f instead of $\tilde{p}$ is that the dilatation (volumetric strain) is a kinematic measure, just like the fluid velocity $\mathbf{v}^f$ (the other nodal degree of freedom) and its associated rate of deformation tensor, whereas $\tilde{p}$ is a function of state (it needs to be formulated as a function of the deformation, just like the viscous fluid stress $\boldsymbol{\tau}$). Here, $\tilde{p}$ is called the *elastic pressure* because it represents the elastic response (equivalent to the stress-strain response in hyperelasticity for

solid mechanics) of the fluid under isothermal conditions. The *linear* model in eq.(4.16.2) is useful for nearly-incompressible flow, typically used for liquids. However, if one expects high volumetric strains in the flow (e.g., at a stagnation point in a high-velocity fluid flow problem), it may be better to employ one of the two *nonlinear* formulations in eq.(4.16.2). In particular, for an ideal gas, the gauge pressure $\tilde{p}$ is given by

$$\tilde{p} = P_r \left(\frac{\theta}{J^f \theta_r} - 1 \right) \quad J^f = 1 + e^f \tag{4.16.3}$$

where θ_r is the (arbitrarily selected) reference temperature and P_r is the corresponding referential absolute pressure (at $\theta = \theta_r$ and $J^f = 1$). When an ideal gas undergoes an isothermal process ($\theta = \theta_r$), eq.(4.16.3) reduces to the *nonlinear* model in eq.(4.16.2), with bulk modulus $K = P_r$. Thus, the *nonlinear* model is consistent with ideal gas law under isothermal processes. Isothermal processes produce zero heat flux, $\mathbf{q} = \mathbf{0}$, however heat is generated by the fluid viscosity at the rate $\boldsymbol{\tau} : \mathbf{D}^f$ where $\mathbf{D}^f$ is the rate of deformation tensor for the fluid (see below). Therefore, to maintain isothermal conditions, we must assume that this heat is implicitly radiated into or out of the fluid domain. For isentropic processes it can be shown that the temperature variation in an ideal gas obeys $\theta/\theta_r = \left(J^f\right)^{1-\gamma}$, where γ is the ratio of isobaric to isochoric specific heat capacities, $\gamma = c_{p0}/c_{v0}$. In that case, the pressure in an ideal gas evolves according to $\tilde{p} = P_r \left(\left(1 + e^f\right)^{-\gamma} - 1 \right)$, which represents a fourth alternative to the three models presented in eq.(4.16.2), with $K = P_r$. Contrary to the simplifying assumptions found in thermodynamic textbooks, an isentropic process is not automatically adiabatic (although the reverse is true). Therefore, when modeling ideal gases undergoing isentropic processes, it is also assumed that the equation of energy balance is implicitly satisfied by suitably radiating heat into or out of the fluid domain.

The fluid pressure $\tilde{p}$ is evaluated from the dilatation e^f (the relative change in volume, or volumetric strain) using the user-specified model from eq.(4.16.2) (the default is *linear*), where $e^f = J^f - 1$ and J^f is the fluid volume ratio. The fluid density ρ^f varies with J^f according to

$$\rho^f = \frac{\rho_r^f}{J^f}\,, \tag{4.16.4}$$

where ρ_r^f is the fluid density in the reference configuration (when the pressure p is zero). The dependence of $\boldsymbol{\tau}$ on the fluid rate of deformation tensor $\mathbf{D}^f = \frac{1}{2}\left(\operatorname{grad} \mathbf{v}^f + \operatorname{grad}^T \mathbf{v}^f\right)$, where $\mathbf{v}^f$ =fluid velocity, is given by a constitutive model which may be chosen from the list provided in Section 4.16.2. Though Newtonian and non-Newtonian fluids may be analyzed in this framework, all fluids are purely viscous (no viscoelasticity is included in this formulation). Setting the viscosity to zero is allowable, for the purpose of analyzing inviscid flow. The user is referred to the *FEBio Theory Manual* for a general description of this isothermal compressible viscous flow framework.

For fluid analyses, which are performed over a fixed mesh, FEBio solves for the components of the fluid velocity $\mathbf{v}^f$ and fluid dilatation e^f at each node. In contrast, fluid-FSI analyses are performed over deforming meshes and FEBio treats fluid-FSI domains as a mixture of a viscous fluid f and a massless, frictionless porous solid s. Thus, the fluid encounters zero resistance as it flows through the deforming mesh (porous solid). The solid constituent of a fluid-FSI domain regularizes the mesh deformation; it is a hyperelastic solid whose constitutive relation is selected by the user; the recommended choice is the unconstrained neo-Hookean solid defined in Section 4.1.4.19, with $\nu = 0$ and E set to a very small (but non-zero) value. For fluid-FSI analyses, FEBio solves for the solid displacement $\mathbf{u}$, the fluid velocity $\mathbf{w}$ relative to the mesh, and the fluid dilatation e^f. The fluid

velocity is then calculated as

$$\mathbf{v}^f = \mathbf{v}^s + \mathbf{w}\,, \tag{4.16.5}$$

where $\mathbf{v}^s$ is the mesh velocity (the material time derivative of the solid displacement $\mathbf{u}$). Since the mesh is fixed in a standard fluid analysis, $\mathbf{v}^s = \mathbf{0}$ and $\mathbf{v}^f = \mathbf{w}$ in that case. Therefore, we use $\mathbf{w}$ to refer to the fluid velocity degrees of freedom for fluid and fluid-FSI analyses. For both types of analyses, the no-slip condition for viscous fluids flowing along a boundary is satisfied by setting $\mathbf{w} = \mathbf{0}$ on that boundary.

On any fluid boundary, the outward surface normal may be denoted by $\mathbf{n}$ and the traction vector on the fluid is given by $\mathbf{t}^f = \boldsymbol{\sigma}^f \cdot \mathbf{n} = -p\,\mathbf{n} + \mathbf{t}^\tau$, where $\mathbf{t}^\tau = \boldsymbol{\tau} \cdot \mathbf{n}$ is the viscous traction. Essential (Dirichlet) boundary conditions may be prescribed on $\mathbf{w}$ and e^f, while natural (Neumann) boundary conditions may be prescribed on $\mathbf{t}^\tau$ and w_n. The appearance of velocity in both essential and natural boundary conditions may seem surprising at first. To better understand the nature of these boundary conditions, it is convenient to separate the velocity into its normal and tangential components, $\mathbf{w} = w_n\mathbf{n} + \mathbf{w}_t$, where $\mathbf{w}_t = (\mathbf{I} - \mathbf{n} \otimes \mathbf{n}) \cdot \mathbf{w}$. In particular, for inviscid flow, the viscous stress $\boldsymbol{\tau}$ and its corresponding traction $\mathbf{t}^\tau$ are both zero, leaving w_n as the sole natural boundary condition; similarly, e^f becomes the only essential boundary condition in such flows, since $\mathbf{w}_t$ is unknown *a priori* on a frictionless boundary and must be obtained from the solution of the analysis.

In general, prescribing e^f is equivalent to prescribing the elastic fluid pressure, since p is only a function of e^f according to eq.(4.16.2). On a boundary where no conditions are prescribed explicitly, we conclude that $w_n = 0$ and $\mathbf{t}^\tau = \mathbf{0}$, which represents a frictionless wall. Conversely, it is possible to prescribe w_n and $\mathbf{t}^\tau$ on a boundary to produce a desired inflow or outflow while simultaneously stabilizing the flow conditions by prescribing a suitable viscous traction. Prescribing essential boundary conditions $\mathbf{w}_t$ and e^f determines the tangential velocity on a boundary as well as the elastic fluid pressure p, leaving the option to also prescribe the normal component of the viscous traction, $t_n^\tau = \mathbf{t}^\tau \cdot \mathbf{n}$, to completely determine the normal traction $t_n^f = \mathbf{t}^f \cdot \mathbf{n}$ (or else t_n^τ naturally equals zero). Mixed boundary conditions represent common physical features: Prescribing w_n and $\mathbf{w}_t$ completely determines the velocity $\mathbf{w}$ on a boundary; prescribing $\mathbf{t}^\tau$ and e completely determines the traction $\mathbf{t}^f = \boldsymbol{\sigma}^f \cdot \mathbf{n}$ on a boundary. Note that w_n and e^f are mutually exclusive boundary conditions, and the same holds for $\mathbf{w}_t$ and the tangential component of the viscous traction, $\mathbf{t}_t^\tau = (\mathbf{I} - \mathbf{n} \otimes \mathbf{n}) \cdot \mathbf{t}^\tau$.

For viscous fluids, the no-slip boundary condition at an impermeable wall is enforced by setting $\mathbf{w}_t = \mathbf{0}$, whereas the wall impermeability condition implies $w_n = 0$. Therefore, these conditions may be combined by prescribing wx, wy and wz components of $\mathbf{w}$ to zero. FEBio offers a range of options for conveniently prescribing natural and mixed conditions on boundary surfaces in fluid analyses (Section 3.13.2).

4.16.1 General Specification of Fluid Materials

The material type for a fluid material is *"fluid"*. The following parameters must be defined:

<density>	Fluid referential density ρ_r	[**M**/**L**^{3}]
<k>	Bulk modulus K	[**P**]
<viscous>	Specification of viscous model for $\boldsymbol{\tau}$	

The <viscous> tag encloses a description of the viscous stress constitutive relation and associated material properties, as may be selected from the list provided in Section 4.16.2. The parameters <density> and <k> must be greater than 0. These parameters are always required.

Example:

```
<material id="1" name="Water" type="fluid">
<density>1</density>
<k>2.2e+09</k>
<viscous type="Newtonian fluid">
<mu>0.001</mu>
<kappa>0</kappa>
</viscous>
</material>
```

4.16.2 Viscous Fluid Materials

Viscous fluid materials provide a constitutive relation for the dependence of the viscous stress $\boldsymbol{\tau}$ on the rate of deformation tensor $\mathbf{D}^f$ of a fluid.

4.16.2.1 Newtonian Fluid

The material type for a Newtonian fluid is "*Newtonian fluid*". The following material parameters must be defined:

<mu>	shear viscosity μ	[**P**·**t**]
<kappa>	bulk viscosity κ	[**P**·**t**]

The viscous shear stress for this material model is

$$\boldsymbol{\tau} = \left(\kappa - \frac{2}{3}\mu\right)(\operatorname{tr}\mathbf{D})\,\mathbf{I} + 2\mu\mathbf{D}$$

Stokes' condition is a constitutive assumption that sets $\operatorname{tr}\boldsymbol{\tau} = 0$, implying that $\kappa = 0$. This assumption is only valid for some fluids (e.g., monoatomic gas [63]). Users may use or ignore this assumption by selecting an appropriate value for κ.

Example:

```
<viscous type="Newtonian fluid">
<mu>0.001</mu>
<kappa>0</kappa>
</viscous>
```

4.16.2.2 Bingham Fluid

The material type for a Bingham fluid [64] is "*Bingham*". The following material parameters must be defined:

<mu>	shear viscosity at infinite shear rate, μ_∞	[**P**·**t**]
<tauy>	yield stress, τ_y	[**P**]
<n>	exponent	

The viscous shear stress for this material model is

$$\boldsymbol{\tau} = 2\mu\mathbf{D}$$

where

$$\mu = \mu_\infty + \frac{\tau_y}{\dot{\gamma}}\left(1 - e^{-n\dot{\gamma}}\right)$$

Here, $\dot{\gamma} = \sqrt{2\mathbf{D} : \mathbf{D}}$ is the engineering shear rate. In the limit as $\dot{\gamma} \to 0$ the viscosity is given by $\mu = \mu_\infty + n\tau_y$. If we define the scalar shear stress $\tau = \sqrt{\boldsymbol{\tau} : \boldsymbol{\tau}/2}$, if follows that $\tau = \mu_\infty\dot{\gamma} + \tau_y\left(1 - e^{-n\dot{\gamma}}\right)$. Effectively, this constitutive model represents a bilinear response for τ versus $\dot{\gamma}$, with a slope of $\mu_\infty + n\tau_y$ when $\tau < \tau_y$ and $\mu = \mu_\infty$ when $\tau > \tau_y$. The exponential function rounds the corner at the intersection of these two lines. For an ideal Bingham fluid one would need to let $n \to \infty$. In practice, values of n between 5 and 50 work well, with the lower end of this range producing faster convergence of the finite element solution.

Example:

```
<viscous type="Bingham">
<mu>1</mu>
<tauy>40</tauy>
<n>40</n>
</viscous>
```

4.16.2.3 Carreau Model

The material type for a Carreau model [25] is "*Carreau*". The following material parameters must be defined:

<mu0>	shear viscosity at zero shear rate, μ_0	[**P**·**t**]
<mui>	shear viscosity at infinite shear rate, μ_∞	[**P**·**t**]
<lambda>	time constant	[**t**]
<n>	power-law exponent	

The viscous shear stress for this material model is

$$\boldsymbol{\tau} = 2\mu\mathbf{D}$$

where

$$\mu = \mu_\infty + (\mu_0 - \mu_\infty)\left(1 + (\lambda\dot{\gamma})^2\right)^{(n-1)/2}$$

Here, $\dot{\gamma} = \sqrt{2\mathbf{D} : \mathbf{D}}$ is the engineering shear rate.

Example:

```
<viscous type="Carreau">
<mu0>0.056</mu0>
<mui>0.0035</mui>
<lambda>3.3</lambda>
<n>0.36</n>
</viscous>
```

4.16.2.4 Carreau-Yasuda Model

The material type for a Carreau-Yasuda model [25] is "*Carreau-Yasuda*". The following material parameters must be defined:

<mu0>	shear viscosity at zero shear rate, μ_0	[**P**·**t**]
<mui>	shear viscosity at infinite shear rate, μ_∞	[**P**·**t**]
<lambda>	time constant	[**t**]
<n>	power-law exponent	
<a>	power-law exponent divider	

The viscous shear stress for this material model is

$$\boldsymbol{\tau} = 2\mu\mathbf{D}$$

where

$$\mu = \mu_\infty + (\mu_0 - \mu_\infty)\left(1 + (\lambda\dot{\gamma})^a\right)^{(n-1)/a}$$

Here, $\dot{\gamma} = \sqrt{2\mathbf{D} : \mathbf{D}}$ is the engineering shear rate.

Example:

```
<viscous type="Carreau-Yasuda">
<mu0>0.056</mu0>
<mui>0.0035</mui>
<lambda>1.9</lambda>
<n>0.22</n>
<a>1.25</a>
</viscous>
```

4.16.2.5 Powell-Eyring Model

The material type for a Powell-Eyring model [25] is "*Powell-Eyring*". The following material parameters must be defined:

<mu0>	shear viscosity at zero shear rate, μ_0	[**P**·**t**]
<mui>	shear viscosity at infinite shear rate, μ_∞	[**P**·**t**]
<lambda>	time constant	[**t**]

The viscous shear stress for this material model is

$$\boldsymbol{\tau} = 2\mu\mathbf{D}$$

where

$$\mu = \mu_\infty + (\mu_0 - \mu_\infty)\,\frac{\sinh^{-1}\lambda\dot{\gamma}}{\lambda\dot{\gamma}}$$

Here, $\dot{\gamma} = \sqrt{2\mathbf{D} : \mathbf{D}}$ is the engineering shear rate.

Example:

```
<viscous type="Powell-Eyring">
<mu0>0.056</mu0>
<mui>0.0035</mui>
<lambda>5.4</lambda>
</viscous>
```

4.16.2.6 Cross Model

The material type for a Cross model [25] is "*Cross*". The following material parameters must be defined:

<mu0>	shear viscosity at zero shear rate, μ_0	[**P**·**t**]
<mui>	shear viscosity at infinite shear rate, μ_∞	[**P**·**t**]
<lambda>	time constant	[**t**]
<m>	exponent	

The viscous shear stress for this material model is

$$\boldsymbol{\tau} = 2\mu\mathbf{D}$$

where

$$\mu = \mu_\infty + (\mu_0 - \mu_\infty)\,\frac{1}{1+(\lambda\dot{\gamma})^m}$$

Here, $\dot{\gamma} = \sqrt{2\mathbf{D}:\mathbf{D}}$ is the engineering shear rate.

Example:

```
<viscous type="Cross">
<mu0>0.056</mu0>
<mui>0.0035</mui>
<lambda>1.0</lambda>
<m>1.0</m>
</viscous>
```

4.16.3 General Specification of Fluid-FSI Materials

The material type for a fluid-FSI material is *"fluid-FSI"*. This type of material is used for fluid flow through a deforming mesh. The material is treated as a special case of fluid-solid mixture, with the solid material used to regularize the mesh deformation. The following parameters must be defined:

<fluid>	Specification of fluid model	
<solid>	Specification of elastic solid model for mesh deformation	

The <fluid> tag encloses a description of the fluid, as given in Section 4.16.1. The <solid> tag encloses a description of the solid, which should be modeled as an unconstrained elastic solid, as given in Section 4.1.4. The recommended choice is the unconstrained neo-Hookean solid defined in Section 4.1.4.19, with $\nu = 0$ and E set to a very small (but non-zero) value.

Example:

```
<material id="1" name="Fluid-FSI domain" type="fluid-FSI">
  <fluid type="fluid">
    <density>1000</density>
    <k>2.2e9</k>
    <viscous type="Newtonian fluid">
      <mu>0.001</mu>
      <kappa>0</kappa>
    </viscous>
  </fluid>
  <solid type="neo-Hookean">
    <density>0</density>
    <E>1e-9</E>
    <v>0</v>
  </solid>
</material>
```

The value of <density> in the <solid> material is internally set to zero regardless of the user-defined value, since the deforming mesh should not be subjected to inertial forces.

4.16.4 General Specification of Biphasic-FSI Materials

The material type for a biphasic-FSI material is *"biphasic-FSI"*. This is a hybrid biphasic material used for modeling dynamic viscous fluid flow and frictional filtration through a porous-deformable solid. In contrast to the standard biphasic material described in Section 4.10, this material accommodates fluid viscosity and fluid dynamics; therefore it can be interfaced seamlessly with a fluid-FSI material to allow viscous fluid transport across that interface. The following parameters must be defined:

<phi0>	solid volume fraction φ_r^s in the reference configuration $(0 < \varphi_r^s < 1)$	[ ]
<fluid>	Specification of the fluid material	
<solid>	Specification of the porous solid material	
<permeability>	Specification of the hydraulic permeability	

The parameter <phi0> must be greater than 0 (no solid) and less than 1 (no porosity). The <fluid> tag encloses a description of the fluid, as given in Section 4.16.1. The <solid> tag encloses a description of the solid matrix constitutive relation and associated material properties, such as those selected from the list provided in Sections 4.1-4.9. The <permeability> tag encloses a description of the permeability constitutive relation and associated material properties, as may be selected from the list presented in Section 4.10.2. To simulate the special case of zero friction between the fluid and solid (such as in the fluid-FSI material), it is permissible to set the permeability to zero.

Example:

```
<material id="2" name="Material2" type="biphasic-FSI">
  <phi0>0.25</phi0>
  <fluid type="fluid">
    <density>1e-9</density>
    <k>1e+09</k>
    <viscous type="Newtonian fluid">
      <mu>1</mu>
      <kappa>0</kappa>
    </viscous>
  </fluid>
  <solid type="neo-Hookean">
    <density>2e-9</density>
    <E>200</E>
    <v>0</v>
  </solid>
  <permeability type="perm-const-iso">
    <perm>0.005625</perm>
  </permeability>
</material>
```

The value of <density> in the <fluid> and <solid> materials represents the true mass density of each constituent.

4.16.5 General Specification of Fluid-Solutes Materials

The material type for a fluid-solutes material is *"fluid-solutes"*. This type of material is used for fluid flow and solute mass transport through a fixed mesh. The following parameters must be defined:

<fluid>	Specification of fluid model	
<osmotic_coefficient>	Specification of osmotic coefficient for fluid-solutes mixture	
<solute>	Specification of (any number of) solute(s)	

The <fluid> tag encloses a description of the fluid, as given in Section 4.16.1. The <solid> tag encloses a description of the solid, which should be modeled as an unconstrained elastic solid, as given in Section 4.1.4. The recommended choice is the unconstrained neo-Hookean solid defined in Section 4.1.4.19, with $\nu = 0$ and E set to a very small (but non-zero) value.

Example:

```
<material id="1" name="Fluid-FSI domain" type="fluid-FSI">
  <fluid type="fluid">
    <density>1000</density>
    <k>2.2e9</k>
    <viscous type="Newtonian fluid">
      <mu>0.001</mu>
      <kappa>0</kappa>
    </viscous>
  </fluid>
  <solid type="neo-Hookean">
    <density>0</density>
    <E>1e-9</E>
    <v>0</v>
  </solid>
</material>
```

The value of <density> in the <solid> material is internally set to zero regardless of the user-defined value, since the deforming mesh should not be subjected to inertial forces.

4.17 Prestrain material

In FEBio a prestrain can be defined for most materials that allows the user to prestrain or prestress the model before loading is applied. This is useful for modeling biologically tissues that exhibit residual strain, or whose initial state cannot assumed to be stress-free (e.g. arteries from in-vivo image data.)

If you use this feature in your published research, please cite the corresponding paper that details the theoretical framework and the implementation in FEBio [55].

4.17.1 Introduction

In the modeling of biological tissues there are various reasons why the model needs to start from a prestressed configuration. For example, the biological tissue may exhibit residual stress or in situ stress. Or the geometry is taken from in vivo data and is subjected to loads in the reference configuration. In these cases, the reference configuration cannot be considered stress-free and this so-called prestress must somehow be accounted for. In large strain analysis, stresses are usually not additive and in addition, it must somehow be ensured that these prestresses are in equilibrium in the reference configuration. In general, this makes prestressing the reference configuration challenging in large strain analyses.

Progress can be made by assuming that the material is hyperelastic and that a prestrain can be found that defines the prestress via the hyperelastic constitutive formulation. Usually this prestrain is characterized by a second-order tensor, termed the prestrain gradient and here denoted by $\boldsymbol{F}_p$. The interpretation of this tensor is that its inverse, applied to a small neighborhood of a point in the reference configuration, would render this neighborhood stress-free. The total elastic response of the material is then defined via the elastic deformation gradient.

$$\boldsymbol{F_e} = \boldsymbol{F} \cdot \boldsymbol{F_p} \tag{4.17.1}$$

Here, $\boldsymbol{F}$is the deformation gradient that is the result of subsequent loading of the reference configuration. Finally, the Cauchy stress and spatial elasticity tangent are given by

$$\boldsymbol{\sigma} = \boldsymbol{\mathcal{F}}\left(\boldsymbol{F}_e\right), \boldsymbol{c} = \boldsymbol{\mathcal{G}}\left(\boldsymbol{F}_e\right) \tag{4.17.2}$$

Here, $\boldsymbol{\mathcal{F}}$ and $\boldsymbol{G}$ are the same response functions from the "natural" material, i.e. the response of the material that starts from a stress-free configuration.

It must be noted that although $\boldsymbol{F}_p$ is termed the prestrain gradient, in general it will not be the derivative of a deformation map. Consequently, a global stress-free reference configuration may not exist and the mapping of a neighborhood in the reference configuration to a stress-free state can at best be defined only locally.

In addition, it must also be recognized that the prestrain gradient is not unique. Due to the requirements of objectivity in the presence of material symmetries, the prestrain gradients $\boldsymbol{F}_p$ and $\boldsymbol{F'}_p = \boldsymbol{F_p Q}$, where $\boldsymbol{Q}$ is any orthogonal tensor, must result in the same material response. This implies that the prestrain can only be dependent on the right stretch tensor $\boldsymbol{V_p}$. However, in general even the right-stretch tensor cannot be defined uniquely. For instance, an isotropic material would only depend on the eigenvalues of this tensor and thus any tensor of the form $\boldsymbol{RDR^T}$, where $\boldsymbol{R}$ is any orthogonal tensor and $\boldsymbol{D}$ is a diagonal tensor with the three eigenvalues of $\boldsymbol{V_p}$on its diagonal, would render the same material response. In the presence of material symmetries, the situation is similar although $\boldsymbol{R}$ will only be part of the symmetry group.

Despite the fact that $\boldsymbol{F}_p$ is not unique, it is far from arbitrary. The most important restriction is that the resulting prestress field $\boldsymbol{\sigma}_p$must be in equilibrium with the prestressed reference configuration. In other words it must hold that

$$div\boldsymbol{\sigma}_p = \boldsymbol{0} \tag{4.17.3}$$

in the interior of the reference domain (in the absence of body forces) and that on the free boundaries of the domain

$$\boldsymbol{\sigma}_p \cdot \boldsymbol{n} = 0 \tag{4.17.4}$$

In a finite element simulation, when these conditions are not satisfied, the mesh will distort and a new reference state is obtained as well as an altered prestrain field. We will denote the deformation gradient between the original (incompatible) reference configuration and prestressed reference configuration by $\boldsymbol{F}_c$. When the mesh distorts the effectively applied prestrain field is also altered. In general, this is not desired and a correction to the prestrain field is necessary that either eliminates the distortion or finds a new reference geometry in which the desired prestrain field is supported.

The prestrain framework as currently implemented assumes that the total effective prestrain that is compatible with the possibly altered prestressed reference configuration is given by the following multiplicative decomposition.

$$\boldsymbol{F}_p = \boldsymbol{F}_c \cdot \hat{\boldsymbol{F}}_p \tag{4.17.5}$$

Both the prestrain gradient and the distortion are in general unknown. The framework implements an iterative algorithm that updates $\boldsymbol{F}_c$ and $\hat{\boldsymbol{F}}_p$ until an effective prestrain gradient is found that is compatible with the possibly distorted reference configuration. Without loss of generality the framework assumes that the altered prestrain gradient $\hat{\boldsymbol{F}}_p$is itself given by the multiplicative decomposition

$$\hat{\boldsymbol{F}}_p = \boldsymbol{G} \cdot \boldsymbol{F}_0 \tag{4.17.6}$$

where $\mathbf{F}_0$ is an initial guess for the prestrain gradient and $\mathbf{G}$ is a correction factor. The algorithm starts by initializing $\mathbf{F}_0$ to a user-defined value and $\mathbf{G}$ to the identity. This initial value can be specified in a variety of ways and the code that generates this initial guess is called the prestrain generator. The framework provides several prestrain generators and users can easily add new ones. Then, a forward analysis is executed keeping $\hat{\mathbf{F}}_p$fixed. Unless the initial guess was compatible with the reference configuration, the mesh will distort. This distortion will define the new value for $\mathbf{G}$ which can be calculated directly from the nodal displacements in the usual manner. Next, the framework will update using a user-defined update rule. How $\mathbf{G}$ is updated depends on the particular application and the framework expects the user do provide the desired update rule. The current implementation provides two particular update rules: one to eliminate the distortion and one to enforce a particular form of the prestrain gradient. See section 3.15.6 on how to use these update rules. In addition, users can easily implement new update rules. After this update, a new forward analysis is executed. This process is repeated until $\mathbf{G}$ converges.

4.17.2 The Prestrain Material

Prestrain can be applied by defining the *prestrain* material. This material acts as a wrapper to the underlying elastic constitutive model. There are two flavors of this material, one for coupled materials and one for uncoupled. In either case, the material defines two properties. The *elastic* property defines the elastic constitutive model. Any of the materials in FEBio could be used, however, the prestrain material has only been validated for hyperelastic materials. The second property, called *prestrain*, defines the prestrain generator. This property defines how the initial prestrain gradient is calculated. Currently, it can be defined as a full second order matrix or as a fiber stretch. The prestrain data can be defined at the material level or at the element level.

The *prestrain* property defines the way that the initial prestrain gradient tensor is calculated. An algorithm that calculates the initial prestrain based on user input is here referred to as a prestrain generator. The type attribute is used to create a specific instance of a prestrain generator. Currently, the following generators are available.

4.17.2.1 prestrain gradient

The *prestrain gradient* generator defines the full prestrain gradient matrix either for the entire material or for each element separately. It has the following parameters.

parameter	description
`ramp`	scale factor that ramps up the prestrain gradient (1)
`F0`	3x3 target prestrain gradient matrix

Comments:

1. The prestrain gradient that is applied is calculated by the following equation.

$$\mathbf{F}_g = \mathbf{I}\,(1-r) + \mathbf{F}_0 r$$

Here, F_g is the applied prestrain gradient, $\mathbf{I}$ is the 3x3 identity tensor, $\mathbf{F}_0$ is the target prestrain gradient, defined by F0 parameter, and *r* is the ramp factor.

Example:

In this example, the prestrain gradient is ramped up to a desired gradient value via a load controller.

```
<material id="1" type="prestrain elastic">
  <elastic type="neo-Hookean">
    <E>1.0</E>
    <v>0.3</v>
  </elastic>
  <prestrain type="prestrain gradient">
    <ramp lc="1">1.0</ramp>
    <F0>2,0,0,0,2,0,0,0,2</F0>
  </prestrain>
</material>
```

4.17.2.2 in-situ stretch

The *in-situ stretch* generator option calculates a prestrain gradient based on a fiber stretch and the fiber vector defined by the elastic material of the prestrain material. This implies that this elastic material must define a *fiber* property.

parameter	description	initial values
stretch	initial fiber stretch	1.0
isochoric	use the isochoric prestrain generator option	1

This option generates one of the following prestrain gradients, depending on the isochoric option.

$$\hat{\mathbf{F}}_{p,iso} = \mathbf{Q}\begin{bmatrix}\lambda & & \\ & \lambda^{-1/2} & \\ & & \lambda^{-1/2}\end{bmatrix}\mathbf{Q}^T, \qquad \hat{\mathbf{F}}_{p,uni} = \mathbf{Q}\begin{bmatrix}\lambda & & \\ & 1 & \\ & & 1\end{bmatrix}\mathbf{Q}^T \tag{4.17.7}$$

If the *isochoric* option is set to 1, then $\hat{\mathbf{F}}_{p,iso}$is used. Otherwise, $\hat{\mathbf{F}}_{p,uni}$is used.

The rotation tensor is defined implicitly via the fiber vector $\boldsymbol{a}$ and the prestrain gradient tensor is effectively determined via,

$$\hat{\mathbf{F}}_{p,iso} = \lambda\mathbf{A} + \mu\,(\mathbf{1}-\mathbf{A})$$

where $\mathbf{A} = \boldsymbol{a}\otimes\boldsymbol{a}$,$\lambda$ the prescribed fiber stretch, and $\mu = \lambda^{-1/2}$ or $\mu = 1$ depending on the *isochoric* option.

Example:

```
<material id="1" type="prestrain elastic">
   <elastic type="coupled trans-iso Mooney-Rivlin">
        <k>0.1</k>
        <density>1e-09</density>
        <c1>0.01</c1>
        <c2>0</c2>
        <c3>0.0139</c3>
        <c4>116.22</c4>
        <c5>535.09</c5>
        <lam_max>1.046</lam_max>
        <fiber type="vector">-0.0804,-0.508,-0.858</fiber>
  </elastic>
  <prestrain type="in-situ stretch">
    <stretch lc="1">1.05</stretch>
    <isochoric>1</isochoric>
  </prestrain>
</material>
```

4.18 Continuous-Discontinuous Damage

In this damage formulation, the damage variable D is embedded inside the strain-energy function. The implementation follows largely the formulation in Balzani [[15]].

The scalar damage variable D is embedded in the strain energy function of the fibers, its relationship is dependent on the Continuous Damage solid constituent chosen. The scalar damage variable D has continuous and discontinuous damage components defined by damage internal damage variables β and γ.

The following material damage parameters must be defined:

<Dmax>	Parameter $D_\infty (0 \leqslant D_\infty \leqslant 1)$
<beta_s>	Parameter β_s $(\beta_s > 0)$
<gamma_max>	Parameter γ_∞ $(\gamma_\infty > 0)$
<t0>	Parameter t_0 $(t_0 \geq 0)$

Note the parameter t_0 is the time in the current loading situation where the damage model is initiated.

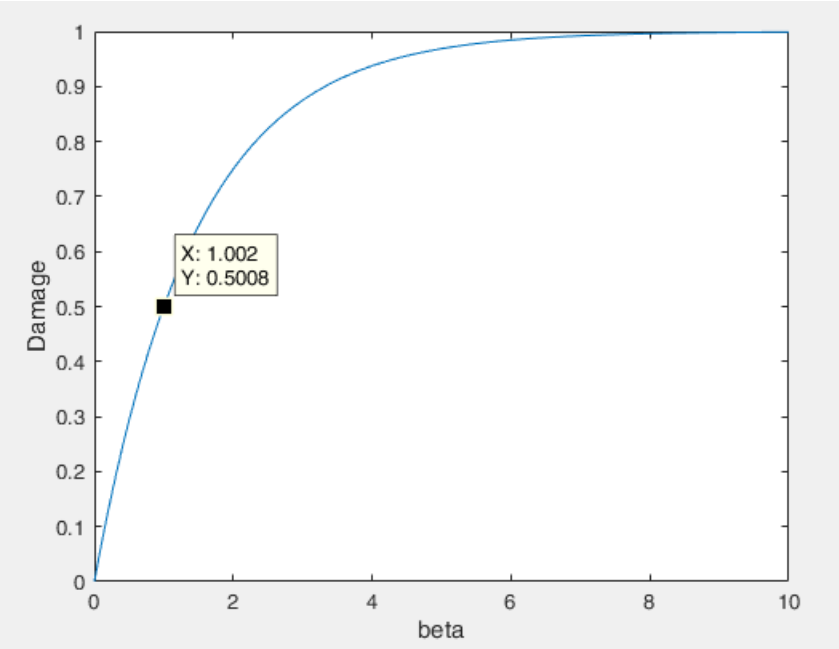
The scalar damage variable D is a function of β, the continuous damage variable, and is given by

$$D(\beta) = D_s \left[1 - \exp\left(\frac{\ln(1-r_s)}{\beta_s}\beta\right)\right],$$

where β is defined by

$$\beta = \left\langle \tilde{\beta} - \tilde{\beta_{ini}} \right\rangle \text{ with } \tilde{\beta} = \int_0^t \dot{\Psi^0} ds,$$

and β_s is the value of the internal material variable β that has reached a certain fraction $r_s (0 \leqslant r_s \leqslant 1)$ of the maximal damage value D_s for a fixed maximum load level. At $\beta = \beta_s$, the overall damage $D = r_s * D_s$. For example, in Figure 1, $D_s = 1$,$r_s = 0.5$,and $\beta_s = 1$, meaning the overall damage in this cycle has reached 50% of the maximal damage value D_s.



Overall Damage variable Das a function of Continuous Damage variable β

To make sure damage evolution begins to accumulate only when damage is accrued, $\tilde{\beta_{ini}}$ is the internal variable $\tilde{\beta}$ at an initial damage state and Ψ^0 is the undamaged effective strain energy density of the fibrils.

D_s is a function increasing the maximally reachable damage value for increased maximum load levels. It is assumed that the maximum damage is itself a function of the internal variable γ, describing the discontinuous damage variable and is given by

$$D_s(\gamma) = D_\infty \left[1 - \exp\left(\frac{\ln(1-r_\infty)}{\gamma_\infty}\gamma\right)\right],$$

where γ is defined by

$$\gamma = \underset{s\epsilon[0,t]}{max} \left\langle \Psi^0 - \Psi^0_{ini} \right\rangle,$$

and γ_∞ is the value of the internal material variable γ that has reached a certain fraction $r_\infty (0 \leqslant r_\infty \leqslant 1)$ of the maximal damage value D_∞ for a fixed maximum load level. At $\gamma = \gamma_\infty$, the overall damage $D_s = r_\infty * D_\infty$ (the same relationship can be adapted from Figure 1 with $\gamma \sim \beta, r_\infty \sim r_s, D_\infty \sim D_s, D_s \sim D$). Ψ^0_{ini} denotes the strain energy density at an initial damage state obtained at the limit of the physiological domain (e.g. in relation to arterial expansion). It is the initial damage strain energy threshold value for the fibers. The time associated with the loading history is denoted by $s\epsilon\mathbb{R}$and the time associated with the current loading situation is represented by $t\epsilon\mathbb{R}^+$.

The strain energy is assumed to be of the following form,

$$\Psi(C, D) = m(P(C, D)),$$

where the modified internal energy function $P(C,D) = (1-D)\bar{P} - c$ with the scalar damage variable D (defined above). $m(P(C,D))$ is a convex and monotonically increasing function, whose first derivative is zero in the origin. Possible functions of $\bar{P}$ are given by

$$\bar{P}_1 := J_4^{(a)}, \bar{P}_3 := K_1^{(a)}, \bar{P}_2 := K_2^{(a)}, \bar{P}_4 := K_3^{(a)},$$

$$(\bar{P}_5 := I_1, \bar{P}_6 := I_2, \bar{P}_7 := I_3),$$

where a denotes the direction of the fibers. In this formulation we take $\bar{P} = \Psi^0$. Note The fifth invariant is not polyconvex, therefore alternative polyconvex invariant functions are defined as

$$K_1^{(a)} = tr\left[\mathrm{Cof}CM_{(a)}\right] = J_5^{(a)} - I_1 J_4^{(a)} + I_2,$$

$$K_2^{(a)} = tr\left[C(I - M_{(a)})\right] = I_1 - J_4^{(a)},$$

$$K_3^{(a)} = tr\left[\mathrm{Cof}C(I - M_{(a)})\right] = I_1 J_4^{(a)} + J_5^{(a)}.$$

The subtraction of c ensures the stress-free reference configuration for the undamaged case is satisfied, c is the value of the function $\bar{P}$ in the natural state and is given by

$$c{=}\bar{P}_i(C = I) \text{ for } i = 1, ..., 7).$$

4.18.1 Damage Fiber Power

The material type for Damage Fiber Power is "*damage fiber power*". The following material parameters must be defined:

<a1>	Parameter α_1 ($\alpha_1 > 0$)
<a2>	Parameter α_2 ($\alpha_2 > 1$)
<kappa>	Parameter κ ($0 \leqslant \kappa \leqslant 2/3$)

By setting,

$$\Psi^0 = \kappa I_1 + \left(1 - \tfrac{3}{2}\kappa\right) K_3$$

and $m(P) = \alpha_1 (P)^{\alpha_2}$,the strain-energy form above can be made suitable for modeling damage,

$$\Psi(C,D) = \alpha_1 \left((1-D)\left[\kappa I_1 + \left(1 - \tfrac{3}{2}\kappa\right) K_3\right] - c\right)^{\alpha_2}.$$

The Cauchy stress takes on the following form,

$$\sigma = \tfrac{2}{J}(1-D)\tfrac{dm}{dP}\left[\kappa b + \left(1 - \tfrac{3}{2}\kappa\right)\left[bI_4 + I_1 I_4 m - I_4 a \bigodot ba\right]\right].$$

Example:

```
<solid type="damage fiber power">
  <a1>1400</a1>
  <a2>2.2</a2>
  <kappa>1e-08</kappa>
  <t0>0.98</t0>
  <Dmax>0.96</Dmax>
```

```
  <beta_s>0.06</beta_s>
  <gamma_max>17.98</gamma_max>
  <fiber type="angles">
    <theta>-39.87</theta>
    <phi>90</phi>
  </fiber>
</solid>
```

4.18.2 Damage Fiber Exponential

The material type for Damage Fiber Exponential is "*damage fiber exponential*". The following material parameters must be defined:

<k1>	Parameter κ_1 ($\kappa_1 > 0$)
<k2>	Parameter κ_2 ($\kappa_2 > 0$)
<kappa>	Parameter κ ($0 \leqslant \kappa \leqslant 1/3$)

The effective strain-energy function is given by,

$$\Psi^0 = \kappa I_1 + (1 - 3\kappa) I_4$$

and

$$m(P) = \tfrac{k_1}{2k_2} \left\{ \exp\left(k_2 \langle P \rangle^2 \right) - 1 \right\},$$

where a denotes the direction of the fibers.
So that,

$$\Psi = \tfrac{k_1}{2k_2} \left\{ \exp\left(k_2 \langle (1 - D_a)\left(\kappa I_1 - (1 - 3\kappa) I_4\right) - 1 \rangle^2 \right) - 1 \right\}.$$

The Cauchy stress then takes on the following form,

$$\sigma = \tfrac{2}{J}(1 - D)\tfrac{dm}{dP}\left[\kappa b + (1 - 3\kappa) I_4 m\right].$$

Example:

```
<solid type="damage fiber exponential">
  <k1>1288.97</k1>
  <k2>400</k2>
  <kappa>0.2</kappa>
  <t0>0.9</t0>
  <Dmax>0.99</Dmax>
  <beta_s>0.001</beta_s>
  <gamma_max>6.67</gamma_max>
  <fiber type="angles">
    <theta>-54.94</theta>
    <phi>90</phi>
  </fiber>
</solid>
```

4.18.3 Damage Fiber exp-linear

This continuous damage formulation uses a slightly modified damage formulation. The total damage is here defined as,

$$D = D_1 + D_2$$

Here, D_1defines the same damage term as defined above. The $D_2$is defined as follows,

$$D_2 = D_3 \left[a \left(\exp \left(b\beta \right) - 1 \right) + c \left(\exp \left(d\beta \right) - 1 \right) \right]$$

The material parameters *a* and *c* control the magnitude of the continuous damage of the two phenomena: 1) slow, constant increased in damage, 2) sharp jump in damage near failure. The other material parameters, *b* and *d*, control the rate of continuous damage accumulation for the two phenomena mentioned above, and is a function of the collagen discontinuous damage. Furthermore,

$$D_3 \left(\gamma \right) = \frac{D_{3\infty}}{1 + exp \left(- \left(\gamma - \gamma_0 \right) / r_\gamma \right)}$$

Here, $D_{3\infty}$ is the maximum achievable discontinuous damage possible for the model, γ_0 determines the shape of the curve, and r_γ controls the rate of discontinuous damage accumulation.

The additional damage parameters to define the D_2 term, are as follows.

D2_a	parameter a
D2_b	parameter b
D2_c	parameter c
D2_d	parameter d
D3_inf	parameter $D_{3\infty}$
D3_g0	parameter γ_0
D3_rg	parameter r_γ

The effective strain-energy function is given by,

$$\Psi^0 = \sqrt{I_4}$$

and

$$m \left(P \right) = \begin{cases} c_3 \left(\exp \left(-c_4 \right) \left(Ei \left(c_4 \left(P + 1 \right) \right) - Ei \left(c_4 \right) \right) - \log \left(P + 1 \right) \right), & P \leqq \lambda^* + 1 \\ c_5 P + c_6 \log \left(P + 1 \right) + d & P > \lambda^* + 1 \end{cases}$$

Here, *d* is determined by requiring continuity at λ^*.

The (elastic) material parameters for this material are as follows.

c3	parameter c_3
c4	parameter c_4
c5	parameter c_5
c6	parameter c_6
lambda	parameter λ^*,the transition between exponential and linear regions.

4.19 First-Order Homogenization

This document describes how to use the first-order homogenization feature in FEBio. Homogenization refers to the process of evaluating the physical response of a macro model by explicitly solving a separate finite element model that represents the micro structure of the material. This micro-model is referred to as the Representative Volume Element (RVE). In the first-order approach it is assumed that the macro and micro scales are separated, which implies that the size of the RVE is negligible compared to the size of the macro model.

4.19.1 Macro model definition

For the most part, the definition of the macro-model is a standard finite element model with properly chosen boundary conditions. The use of first-order homogenization is controlled entirely by the material definition.

To use the first-order homogenization formulation use the “micro-material” material. This material has the following parameters.

parameters	**description**	**default**
RVE	The file name of the RVE model (1)	N/A
rve_type	The type of RVE model (2)	0
bc_set	The name of the nodeset (in the RVE) that defines the outside surface (3)	(optional)
probe	Defines a point in the macro model where the RVE will be tracked. (4)	(none)

Comments:

1. The RVE parameter defines the file name of the RVE model. The RVE is a standard FE model, although somewhat slimmed down. See the rve_type parameter definition and the section RVE model definition below.

2. This parameter determines the type of RVE (and its assumed boundary conditions) that will be used and must be one of the following values.

rve_type	**description**
0	Prescribed displacement RVE: The position of all RVE boundary nodes is fully prescribed
1	Periodic RVE: RVE with periodic boundary constraints. Constraints are enforced with linear constra

3. The bc_set defines the name of the nodeset (defined in the RVE model), that defines the outside surface. In general, this parameter does not need to be defined. It is only needed for problems using rve_type = 0, and for which the RVE geometry is not a closed cube.

4. By default, FEBio will only output the deformation of the macro model to the plot file. It will not output any results from the RVE models. (Given there could be thousands of RVE models, this would be too cumbersome to do.) However, users can request FEBio to output the deformation of the RVE model at specific points in the model using the probe option. The probe tag requires the following child tags.

probe parameter	**description**
element_id	the ID of the element to probe
gausspt	the index of the integration point
file	output file name for RVE model

4.19.2 RVE model definition

The RVE model is a standard FE model, with a few caveats.

- The time stepping parameters will be ignored, since the macro model controls the time stepping of the RVE to make sure that both models advance in sync.
- The boundary conditions of the RVE depend on the rve_type parameter:
  - rve_type = 0: The RVE model should not have any boundary conditions applied. The RVE may define a nodeset that defines the outside surface of the RVE. This is generally not needed, but must be defined when the RVE geometry is not a closed cube.
  - rve_type = 1: The RVE should have periodic constraints applied for all opposing surfaces.
- The RVE model does not need an Output section, but may define the plotfile section. This will be used to define the conents of the probe's plotfiles.
- The size of the RVE is assumed to be much smaller than the macro model. It is allowed to use a different length unit for the coordinates of the RVE nodes. However, the unit of stress must be the same for both the macro model and the RVE model.

Chapter 5

Restart Input file

5.1 Introduction

Users can request FEBio to output a binary dump file. This dump file can be used to restart the analysis from the time point saved in the dump file. The user can restart the analysis either directly from this binary dump file or via a restart input file, which is described in this chapter. See section 2.8.4 for more information on how to use the restart feature.

This chapter describes the format of the restart input file. This file is used to redefine some parameters when restarting a previously terminated run and can also be used to extend the analysis. The structure is very similar to the FEBio input file and also uses XML formatting. Since the file uses XML, the first line must be the XML header:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

The next line contains the root element of the restart file, and has to be:

```
<febio_restart version="2.0">
```

The restart file is composed of the following sections. These sections are sub-elements of the *febio_restart* root element.

Archive define the binary dump file used for restarting.

Control redefine some control parameters.

LoadData redefine or add some loadcurves.

Step add an analysis step.

All sections are optional except for the Archive section. In the following paragraphs we describe the different sections in more detail.

5.2 The Archive Section

The Archive section must be the first sub-element of the *febio_restart* root element. This section defines the name of the binary dump file:

```
<Archive>archive.dmp</Archive>
```

This file will be created by FEBio when requested by the user and contains the solution of the analysis up to the last converged time step.

5.3 The Control Section

The following control parameters can be redefined:

Parameter	Description
dtol	convergence tolerance for displacements
etol	convergence tolerance for energy
rtol	convergence tolerance for residual
lstol	line search tolerance
max_refs	maximum number of stiffness reformations
max_ups	maximum number of BFGS updates
restart	restart file generation flag
plot_level	defines the frequency of the plot file generation

5.4 The LoadData Section

In the LoadData section some or all of the load curves can be redefined, or new load curves can be added. The syntax is identical to the LoadData section of the FEBio input file:

```
<LoadData>
  <loadcurve id="1">
    <loadpoint>0, 0</loadpoint>
    ...
    <loadpoint>1, 0.54</loadpoint>
  </loadcurve>
</LoadData>
```

In this case, the loadcurve *id* is the loadcurve number of the loadcurve that the user wishes to redefine.

New loadcurves can be added as well. This is useful when in addition adding new Step sections, where new boundary conditions are defined. When adding new loadcurves, make sure to continue the numbering of the parent input file. That is, if the last loadcurve of the parent input file has and id of n, then the first new load curve defined here must have id of n+1.

5.5 The Step Section

The Step section can be used to add additional analysis steps to the model. These steps are executed after the initial model has completed. This section cannot be used to modify the steps defined in the parent file. See chapter 6 for more information on the Step section or multi-step analyses in general.

Using the Step section, additional boundary conditions, loads, contact definitions, etc. can be added, just like in a regular model input file. The syntax is the same as the Step section in the regular model input file. The only difference is that the type attribute is required to define the type of analysis. The type has the same role as the Module section in the regular FEBio input file.

5.6 Example

5.6.1 Example 1

The following example defines a restart input file. No parameters are redefined. Only the mandatory *Archive* element is defined. In this case the analysis will simply continue where it left off:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<febio_restart version="1.0">
  <Archive>out.dmp</Archive>
</febio_restart>
```

5.6.2 Example 2

In the following example an additional step is defined, which defines a new boundary condition and a new loadcurve.

```
<?xml version="1.0" encoding="utf-8"?>
<febio_restart version="2.0">
  <Archive>out.dmp</Archive>
  <Step type="solid">
    <Control>
      <time_steps>10</time_steps>
      <step_size>0.1</step_size>
    </Control>
    <Boundary>
      <prescribe bc="x" node_set="set1">
         <scale lc="2">1.0</scale>
      </prescribe>
    </Boundary>
  </Step>
  <LoadData>
    <loadcurve id="2" type="linear">
      <point>1, 0</point>
      <point>2, 1</point>
    </loadcuve>
  </LoadData>
</febio_restart>
```

Notice the “type” attribute in the Step section. This serves a similar purpose as the Module section in the FEBio input file and defines the solver that FEBio will use to solve this step. This attribute is required when defining steps in the restart file. (The Module section is not supported in the restart input file.)

The prescribed boundary condition references a node set named “set1”. This node set must be defined in the parent input file. No new node sets (or surfaces, etc.) can be defined in the restart input file.

Chapter 6

Multi-Step Analysis

A multi-step analysis is defined using multiple steps, where in each step the user can redefine control parameters, boundary conditions, loads, contact interfaces, and other model components. This is useful, for instance, for defining time-dependent boundary conditions or for switching between different analysis types during the simulation.

Multi-step models require a slightly different file organization compared to single-step models. More specifically, the Control section is not specified at the top of the input file. Instead, a ***Steps*** section is added at the bottom of the file, which contains a list of **step** sub-sections for each step. Each step then defines its own *Control* section, boundary conditions, loads, etc.

6.1 The Step Section

The multi-step analysis feature organizes the input file in a slightly different way: the ***Control*** section is no longer defined at the top of the file, and instead, a new ***Steps*** section is added to the bottom of the file, which will contain a list of analysis steps. Each analysis step requires its own ***step*** sub-section. In this step section, the user can redefine the control section, boundary section, loads, contact, and other model components. The following format is suggested when defining a multi-step analysis:

```
<febio_spec version="4.0">
  <Module type="solid"/>
  <Material>
    <!-- materials go here -->
  </Material>
  <Mesh>
    <!-- mesh definition goes here -->
  </Mesh>
  <MeshDomains>
    <!-- mesh domains goes here -->
  </MeshDomains>
  <Boundary>
    <!-- global boundary conditions -->
  </Boundary>
  <LoadData>
    <!-- load curve data goes here -->
```

```
  </LoadData>
  <Steps>
    <step>
      <Control>
        <!-- local control settings -->
      </Control>
      <Boundary>
        <!-- local boundary conditions -->
      </Boundary>
    </step>
  </Steps>
</febio_spec>
```

The first part of the file looks similar to a normal input file, except that the control section is not specified. Also, the Boundary, Loads, Constraints, Contact, sections should only contain global boundary conditions, i.e. boundary conditions that will remain active in all analysis steps.

At the end of the file the user defines as many **step** sections as needed. In each **step** section, the user can now define the control parameters and model components.

The sub-sections that can be defined in a step are:

Control Defines the steps control parameter, such time stepping, solver settings, etc.

Initial Applies initial conditions that take effect at the start of the step.

Boundary Defines boundary conditions that are only applied in this step.

Loads Defines loads that are only applied in this step.

Constraints Defines constraints that are only applied in this step.

Rigid Defines rigid constraints, loads, and connectors in this step.

MeshAdaptor Defines mesh adaptors that will be applied in this step.

6.1.1 Boundary Conditions

In a multi-step analysis boundary conditions can be applied that are only active during the step. This applies to the *Boundary*, *Loads*, *Contact* and *Constraints* section of the FEBio input file. These sections can thus be placed inside the *Step* section to define a boundary condition that is only active during the step. If one of these sections is defined before the first *Step* section, the corresponding boundary conditions remain enforced during all the steps.

6.1.2 Relative Boundary Conditions

Some boundary conditions can be defined as relative boundary conditions. This means that the corresponding conditions will be applied to the final configuration of the previous step and not to the original reference configuration. For example, if a prescribed displacement is defined as relative the displacement will be taken with respect to the positions of the final configuration of the previous step. This makes it possible, for example, to switch between load and displacement controlled boundary conditions in multi-step analyses.

6.2 An Example

The following example illustrates the use of the multi-step feature of FEBio. This problem defines two steps. In the first step, a single element is stretched using a prescribed displacement boundary condition. In the second step, the boundary condition is removed and the analysis type is switched from quasi-static to a dynamic analysis. Note the presence of the global fixed boundary constraints, which will remain enforced during both steps:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<febio_spec version="4.0">
<Module type="solid"/>
<Material>
<material id="1" name="Material 1" type="neo-Hookean">
<density>1.0</density>
<E>1</E>
<v>0.45</v>
</material>
</Material>
<Mesh>
<Nodes>
<node id="1">-2.0,-0.5, 0.0</node>
<node id="2">-2.0,-0.5, 1.0</node>
<node id="3">-2.0, 0.5, 0.0</node>
<node id="4">-2.0, 0.5, 1.0</node>
<node id="5"> 2.0,-0.5, 0.0</node>
<node id="6"> 2.0,-0.5, 1.0</node>
<node id="7"> 2.0, 0.5, 0.0</node>
<node id="8"> 2.0, 0.5, 1.0</node>
</Nodes>
<Elements type="hex8" name="Part1">
<elem id="1">1,5,7,3,2,6,8,4</elem>
</Elements>
<NodeSet name="set1">
<node id="1"/>
<node id="2"/>
<node id="3"/>
<node id="4"/>
</NodeSet>
<NodeSet name="set2">
<node id="1"/>
<node id="2"/>
<node id="5"/>
<node id="6"/>
</NodeSet>
<NodeSet name="set3">
<node id="1"/>
<node id="3"/>
<node id="5"/>
```

```
<node id="7"/>
</NodeSet>
<NodeSet name="set4">
<node id="5"/>
<node id="6"/>
<node id="7"/>
<node id="8"/>
</NodeSet>
</Mesh>
<MeshDomains>
<SolidDomain name="Part1" mat="Material 1"/>
</MeshDomains>
<Boundary>
<bc type="zero displacement" node_set="set1">
<dofs>x,y,z</dofs>
</bc>
</Boundary>
<LoadData>
<load_controller id="1" type="loadcurve">
<points>
<point>0,0</point>
<point>1,0.1</point>
</points>
</load_controller>
</LoadData>
<Step>
<step id="1">
<Control>
<time_steps>10</time_steps>
<step_size>0.1</step_size>
</Control>
<Boundary>
<bc type="prescribed displacement" node_set="set4">
<dof>x</dof>
<scale lc="1">1.0</scale>
</bc>
</Boundary>
</step>
<step id="2">
<Control>
<time_steps>50</time_steps>
<step_size>0.5</step_size>
<analysis>DYNAMIC</analysis>
</Control>
</step>
</Step>
</febio_spec>
```

Chapter 7

Parameter Optimization

This chapter describes FEBio's parameter optimization module. This module tries to estimate model parameters by solving an inverse finite element problem, where the "solution" of the problem is known, and the problem's model parameters are sought. For instance, the solution can be an experimentally determined reaction force curve, and the unknown parameters are the material parameters that will recreate the reaction force curve by solving a forward FE problem. Another common application is determining the model parameters that achieve a desired load, displacement, or other goal.

In any case, the opimization module tries to minimize an objective function of the form,

$$\varphi(\mathbf{a}) = \sum_{i=1}^{n} [y_i - f(x_i; \mathbf{a})]^2 .$$

Here, the (x_i, y_i) are user-defined data pairs and $f(x; \mathbf{a})$is the function that extracts the corresponding data from the model. The optimization module tries to find the model parameters **a** that minimize the function φ. It does this by repeatedly evaluating the function f, which will usually call FEBio to solve a forward FE problem.

The optimization module requires a separate input file that describes all the information needed for solving an optimization problem. This input file is described next.

7.1 Optimization Input File

Like all FEBio input files, the parameter optimization input file is an xml-formatted text file. The following sections are defined:

Task optional section defining the task that is executed to solve the FE problem.

Options optional section defining the optimization control parameters.

Objective defines the objective function that will be minimized.

Parameters defines the model parameters that are to be determined.

Constraints defines linear constraints for the model parameters (requires a solution method that supports linear constraints, e.g. constrained levmar).

In the following paragraphs each section will be explained in detail.

7.1.1 Task Section

The optional Task section can be used to define the task that is executed by FEBio when solving the FE model. By default, FEBio will solve the standard forward problem, but if needed, a custom task can be defined.

7.1.2 Options Section

This section defines the control parameters for the optimization. The options that can be defined will depend on the chosen optimization method, which is set with an attribute of the Options section. The following table shows the currently supported optimization algorithms.

type	Description
levmar	Levenberg-Marquardt
constrained levmar	Levenberg-Marquardt with linear constraints (1)
powell	Powell's method
scan	Parameter domain scanning (2)

Comments:

1. The "constrained levmar" uses the third-party levmar library http://users.ics.forth.gr/~lourakis/levmar/. For additional information please see the library's home page.
2. The "scan" method will iterate over the parameter range in increments defined by the fourth parameter value defined in the Parameter section. It returns the parameters that resulted in the lowest objective value of visited points, but it may not be the actual minimum.

Note that the Options section is optional. When omitted, the levmar method is used with default values for the control parameters.

The following parameters can be defined.

Parameter	Description	Default
obj_tol	Convergence tolerance for objective function. (1)	0.001
f_diff_scale	Forward difference scale factor (2)	0.001
log_level	Sets the amount of output that is generated on screen and in the logfile by the FEBio solver (3)	LOG_NEVER
tau	Step size scale factor (4)	1.0e-3
print_level	Sets the amount of output generated by the optimization module (5)	PRINT_ITERATIONS

For example,

```
<Options type="constrained levmar">
  <obj_tol>0.001</obj_tol>
  <f_diff_scale>0.001</f_diff_scale>
  <print_level>PRINT_ITERATIONS</print_level>
</Options>
```

Comments:

1. The objective function that is to be minimized is a function of the form:

$$\varphi(\mathbf{a}) = \sum_{i=1}^{n} [y_i - f(x_i; \mathbf{a})]^2 .$$

   Here, $f(x; \mathbf{a})$ is the function that describes the model, $\mathbf{a}$ is a vector with the (unknown) model parameters and the (x_i, y_i) are the user-defined data that approximates the ideal model.

2. The optimization method currently implemented requires the calculation of the gradient of the model function f with respect to the model parameters $\mathbf{a}$. Since this gradient is not known, it will be approximated using forward differences. For example, the k-th component of the gradient is approximated as follows.

$$\frac{\partial f}{\partial a_k} \approx \frac{1}{\delta a_k} [f(a_1, \cdots, a_k + \delta a_k, \cdots, a_m) - f(a_1, \cdots, a_k, \cdots, a_m)]$$

   The value for $\delta a_k$is determined from the following formula.

$$\delta a_k = \varepsilon (1 + a_k)$$

   where, $\varepsilon$is the forward difference scale factor which can be set by the user with the *fdiff_scale* option.

3. The *log_level* allows the user to control exactly how much output from the time iterations of each optimization iteration is written to the screen and logfile. The following values are allowed:

Value	**Description**
LOG_DEFAULT	Use default settings (may be omitted)
LOG_NEVER	Don't generate any output
LOG_FILE_ONLY	Save to the logfile only
LOG_SCREEN_ONLY	Only print to the screen
LOG_FILE_AND_SCREEN	Save to the logfile and print to the screen

4. The step size scale factor *tau* can be set as an input only for the Constrained Levenberg-Marquardt method. It scales the initial guess for the damping parameter.

5. The *print_level* sets how much output is generated during the optimizations. In particular, it controls if the objective function values are printed are not. The following values are allowed:

Value	**Description**
PRINT_ITERATIONS	Print minimal iterations info
PRINT_VERBOSE	Print a lot of information, incl. objective function values

7.1.3 Parameters Section

This section defines the material parameters that are to be determined. Each parameter is defined using the *param* element:

```
<Parameters>
  <param name="[name]">[guess], [min], [max], [scale]</param>
</Parameters>
```

The *name* attribute gives the name of the parameter that is to be determined. The following section describes the format to reference a model parameter. Each parameter takes four values: [guess] is the initial guess for this parameter, [min] and [max] are the minimum and maximum values respectively for this parameter, and [scale] is a representative scale (magnitude) for this parameter (although the precise interpretation of this parameter depends on the particular solver). This value is used to normalize the optimization parameter and improve convergence. If not specified by the user, it defaults to the initial guess.

For example, to optimize the material parameters of a neo-Hookean solid, named *mat1,* use the following syntax.

```
<Parameters>
  <param name="fem.material('mat1').E">1.0, 0.0, 5.0, 1.0</param>
  <param name="fem.material('mat1').v">0.1, 0.0, 0.5, 1.0</param>
</Parameters>
```

This example defines two material parameters. The first component of the name (here *mat1*) is the name of the material as defined in the model input file. The second component (here *E* and *v*) are the names of the material parameters. See Appendix B for more information on how to reference model parameters.

7.1.4 Objective Section

This section defines the functional model that is used to evaluate the objective function (i.e. the function f above). The particular functional model is defined via the *type* attribute, and the following table lists the available options.

Value	Description
data-fit	Fits data, usually specified as a function of time, to a model output parameter
target	Optimizes until one or more output model parameters achieve a target value.
element-data	The target data is defined as a tabulated list of element values.
node-data	The target data is defined as a tabulated list of nodal values.

These options will be described separetely in the following sections.

7.1.4.1 The data-fit model

The *data-fit* model tries to fit predicted FE data to a user-defined data curve of time-value pairs. The data curve usually represents some experimentally obtained result, e.g. reaction force, as a function of time. The data-fit model needs a data source, i.e. a function for extracting the model data, and the data curve.

```
<Objective type="data-fit">
  <fnc type="[enter type]">
    ...
  </fnc>
  <data>
    <pt>0, 0</pt>
    ...
    <pt>1, 1</pt>
  </data>
</Objective>
```

The data source is defined by the type attribute of the fnc tag. There are currently two supported values.

- *parameter*: The function will track the time evolution of a model parameter.
- *filter_positive_only*: Another data source is needed to extract model data (e.g. parameter), but only positive values are returned. If the model parameter's value was negative, zero will be returned by the function.

In the case of *parameter*, a child element defines the parameter that will act as the data source for the objective function. (See Appendix B on more information on referencing model parameters.)

```
<fnc type="parameter">
  <param name="fem.rigidbody('rigid').Fz"/>
</fnc>
```

This defines the data function as a function of time. You can also change the ordinate of the function to create other functional dependencies. For example, the following creates a load-displacement curve for a rigid body.

```
<fnc type="parameter">
  <ordinate name="fem.rigidbody('rigid').position.x"/>
  <param name="fem.rigidbody('rigid').Fz"/>
</fnc>
```

As another example, the following creates a stress-strain curve for a single solid element, using the left Hencky (logarithmic) strain component along the z-axis (denoted by *hz*), and the Cauchy normal stress component along z-axis (denoted by *sz*) in element number 1,

```
<fnc type="parameter">
  <ordinate name="fem.element_data('hz',1)"/>
  <param name="fem.element_data('sz',1)"/>
</fnc>
```

7.1.4.2 The target model

The *target* model tries to optimize model parameters such that one or more output variables achieve a user-defined target value. This model requires a list of variables and target values for each variable.

```
<Objective type="target">
  <var name="[enter name]">[target value]</var>
  ...
</Objective>
```

The name refers to a model output parameter. The example below will optimize until the X-euler angle of the rigid body 'Material2' becomes 0.5.

```
<Objective type="target">
  <var name="fem.rigidbody('Material2').euler.y">0.5</var>
</Objective>
```

7.1.4.3 The element-data model

The *element-data* model uses a tabulated list of element values as the target for the optimization. In addition, it requires an output variable that will be matched against the element data.

```
<Objective type="element-data">
  <var type="[enter variable name]"/>
  <data>
    <elem id="1">0.123</elem>
    ...
  </data>
</Objective>
```

The variable can be any of the element output data listed in the logfile section (See 3.6.6).

7.1.4.4 The node-data model

The *node-data* model uses a tabulated list of nodal values as the target for the optimization. In addition, it requires an output variable that will be matched against the nodal data.

```
<Objective type="node-data">
  <var type="[enter variable name]"/>
  <data>
    <node id="1">0.123</elem>
    ...
  </data>
</Objective>
```

The variable can be any of the node output data listed in the logfile section (See 3.19.1.1).

7.1.5 Constraints Section

The Constrained Levenberg-Marquardt method allows linear constraints on the material parameters. If $\mathbf{a}$ is the material parameter vector, then the linear constraint is of the form:

$$c_1 a_1 + c_2 a_2 + ... + c_n a_n + b = 0 \quad .$$

The coefficients $c_1, c_2, ..., c_n, b$ are the inputs of the constraint tag. For example, if the linear constraint is $2a_1 - a_2 + 3 = 0$, then the Constraints section would be:

```
<Constraints>
  <constraint>2, -1, 3</constraint>
</Constraints>
```

7.2 Running a Parameter Optimization

As explained above, a parameter optimization problem is described using two input files. First, a standard FEBio input file that defines the mesh, materials, boundary conditions, etc. The second input file describes the parameter optimization problem, such as the objective and which model parameters are to be optimized. The format of the second file is described above. A parameter optimization can only be initiated from the command line. For example:

```
>febio -i model.feb -s optim.feb
```

The output of a parameter optimization analysis is a log file that contains the screen output of the FEBio run as well as the optimized parameter values.

7.3 An Example Input File

Below follows a complete example of an optimization input file.

```
<?xml version="1.0"?>
<febio_optimize version="2.0">
  <Options>
    <obj_tol>0.001</obj_tol>
    <f_diff_scale>0.001</f_diff_scale>
  </Options>
  <Parameters>
    <param name="fem.material[0].E">1, 0, 5</param>
    <param name="fem.material[0].v">-0.5, 0, 0.5</param>
  </Parameters>
  <Objective type="data-fit">
    <fnc type="parameter">
      <param name="fem.rigidbody('rigid').Fz"/>
    </fnc>
    <data>
      <point>0.0, 0</point>
      <point>0.5, 1</point>
      <point>1.0, 2</point>
    </data>
  </Objective>
</febio_optimize>
```

Comments:

1. Notice that the xml root element is *febio_optimize* for the optimization input file. The version number has to be 2.0 (the 1.0 version is no longer supported).

2. The FEBio input file that contains the actual FE model data has to defined on the command line using the -i command option. This file is a standard FEBio input file that defines all geometry, materials, boundary conditions and more.

3. The initial values of the parameters are defined in the optimization input file. The values in the model input file are ignored.

4. The *Options* section is included here, but can be omitted. If omitted default values will be used for all control parameters.

Chapter 8

Troubleshooting

Running a nonlinear finite element analysis can be a very challenging task. The large deformations and complex constitutive models can make it very difficult to obtain a converged solution. There are many causes for nonconvergence, ranging from element inversions, material instability, failure to enforce contact constraints and many more. Sometimes it is possible that FEBio gives you a converged solution, but the solution is meaningless or at least not what was expected.

Fortunately, many of these issues can be prevented or solved with little effort. This chapter discusses some strategies to prevent common problems and to troubleshoot a problematic run. It offers some sanity checks before you run your model which can save you a lot of frustration down the road. And when things do go bad, we hope that the strategies suggested here may prove helpful.

However, keep in mind, that the finite element method cannot solve all problems. It is a very powerful numerical method, but with limitations. Understanding these limitations and how they affect your modeling work is crucial in becoming a good analysist.

8.1 Before You Run Your Model

In this section we'll discuss what a well-defined finite element model is and some things you may need to check before you run your model in FEBio.

A well-defined finite element model contains a finite element mesh, a valid material and properly defined boundary and contact conditions in order to define a unique solution to the problem under study. We will look at each of these requirements in more detail in the following sections.

8.1.1 The Finite Element Mesh

A finite element mesh is required to solve a problem with FEBio. The mesh defines a discretization of the problem domain in nodes and connected elements. FEBio only supports certain elements and thus the mesh must be composed of elements from this set. See Section 3.6.2 for a discussion of the supported elements. In addition, FEBio assumes a specific ordering of the nodes of an element. FEBio cannot discover if the nodes are in the correct order, but if they are not, FEBio will most likely have trouble converging or throw negative jacobians. If FEBio discovers negative jacobians before the first time step, it is likely that the nodes of the elements are not defined in the proper order.

For shell elements, the initial thickness of an element is also important. When elements are too thick (the thickness is of the same order as the element size), FEBio may complain about negative

jacobians. This may be particularly a problem in areas of high curvature. The only solution around this issue might be to remesh the problematic area.

8.1.2 Materials

A material in FEBio defines the constitutive response of the domain to which the material is assigned. See Chapter 4 for a detailed list of all the available materials in FEBio. It is important to understand that the module defines which materials you can use. For example, the biphasic material cannot be used in the *solid* module. Although some cases of invalid material use are caught, in many situations the resulting behavior is undefined. The following table shows a list of some of FEBio's special materials and the modules in which they can be used.

Material	Solid	Biphasic	Solute	Multiphasic	Heat
biphasic		**YES**	**YES**	**YES**	
biphasic-solute			**YES**	**YES**	
triphasic			**YES**	**YES**	
multiphasic				**YES**	
isotropic Fourier					**YES**

In addition to using the proper materials for a given module, it is also important to understand that most material parameters have a limited range in which they define valid constitutive behavior. For example, in a neo-Hookean material the Young's modulus must be positive and the Poisson's ratio must larger than -1 and less than 0.5. FEBio has most of these limits implemented in the code and will throw an error when a parameter value is defined that falls outside the valid range.

Since material parameters can be defined as functions of time through a loadcurve, FEBio will check all parameters at the beginning of each time step. When a parameter has become invalid the run will be aborted.

Some constitutive models are only valid for a limited amount of deformation. All materials in FEBio are designed for large deformation (in the sense that they are objective under arbitrary deformation), but that does not imply unlimited deformation. When the deformation becomes too large, the material response may become unphysical and although FEBio gives an answer, the result may be meaningless (see Section 8.9 for a discussion on interpreting the result). Some materials also become unstable at extremely large deformations. A classical example is the Mooney-Rivlin material. For a certain range of values of the material parameters, and under certain loading conditions, the stress-strain response can have a zero slope at large deformations. In that case, FEBio will most likely not be able to converge since the slope of the stress-strain response is used to progress towards the solution.

8.1.3 Boundary Conditions

Boundary conditions are necessary to uniquely define the solution of the problem under study. Improperly defined boundary conditions can lead to underconstrained or overconstrained problems.

If the problem is underconstrained the solution is not uniquely defined and FEBio will not be able to find a solution. The most common example of an undercontrained problem is one where the rigid body modes are not constrained. A rigid body mode is a mode of deformation that does alter the stress in the body. Affine translation and rotation of the entire mesh are two ways to introduce a rigid body mode. In a (quasi-)static finite element analysis all rigid body modes must

be properly constrained in order to find a unique solution. This can often very easily be achieved by constraining and edge or face of the model from moving at all.

If the problem is over-constrained, FEBio will usually find an answer but it is probably not the solution that was sought. The most common cause of an over-constrained model is one that has conflicting boundary conditions. For example, a force is applied to a node that is fixed.

8.2 Debugging a Model

When a model is not running as expected, the first thing to try is rerun the problem using FEBio's debug mode. Debug mode can be activated by adding a –g on the command line. For example,

```
>febio -i file.feb -g
```

When in debug mode, FEBio will do additional checks during the solution (e.g. check for NAN's, check for zeroes on the diagonal of the stiffness matrix, etc.) and will store all non-converged states to the plot file. Inspecting these non-converged states can often be useful to identify the cause of the problem. (E.g. when the model seems to disappear at some point might indicate a rigid body mode.)

Since storing all non-converged states may make the plot-file too large for post-processing, debug mode can also be turned on during the analysis. To do this, start the problem normally (without the debug flag –g). Then, right before the problem starts to fail, interrupt the run (using ctrl+c) and wait for the FEBio prompt to appear. Once it appears, enter *debug* [ENTER], and then *cont* [ENTER] to continue the run in debug mode. Alternatively, a break point can be specified on the command line or at the FEBio prompt, which will instruct FEBio to break at a particular point in the analysis. At that point, the febio prompt will appear, and the debug mode can be toggled. For more information on setting breakpoints, consult section 2.8.3.

Another option is to use the -g2 flag, instead of the -g flag. This flag will do all of the debug checks, but only output the non-converged state that caused febio to fail to the plot file.

8.3 Common Issues

In this section we take a look at some of the most common problems you may run into when solving a finite element problem with FEBio. We'll discuss possible causes of and solutions to these issues.

8.3.1 Inverted elements

When an element inverts, the element becomes so distorted that in certain areas of the element the Jacobian becomes zero or negative. In order to obtain a physically realistic solution, the Jacobian of the deformation must be positive at all points of the domain. Thus, when a negative Jacobian is found in an element, FEBio cannot continue. Fortunatly, FEBio's automatic time-stepping algorithm can usually circumvent this problem by cutting the time step back and retrying the step using a smaller time step. In addition, FEBio will recalculate the global stiffness matrix. The combination of a reduced time step and a refactored stiffness matrix will often be sufficient to overcome the negative Jacobian. However, when it is not, you may have run into a more serious problem. These are some of the common causes that may require additional user intervention.

By default, when FEBio encounters inverted elements, it will print an error message stating that “negative jacobians” were detected. You can request FEBio to print out a detailed list of the inverted elements, by setting the *output_negative_jacobian* configuration flag in the FEBio configuration file. (see 9)

8.3.1.1 Material instability

At large deformations, some materials can become unstable. This is usually caused by a softening of the material at large deformations. When the material softens too much the global stiffness matrix can become ill-conditioned and FEBio will not be able to find the correct solution. Unfortunately, there is no easy solution around this issue and you may need to consider using a different constitutive model.

8.3.1.2 Time step too large

Altough the auto-time stepper will automatically adjust the time step in case of trouble, it is designed to work within user defined limits and sometimes they are not set properly to solve the problem. The most common issue is that the minimum time step is set too large. Cutting back the minimum in addition to the initial time step can often help in preventing negative Jacobians.

8.3.1.3 Elements too distorted

When elemens get too distorted they might not be able to deform any further without inverting the element. In this case, it may be necessary to adjust the mesh in the area where the elements are inverting.

8.3.1.4 Shells are too thick

A shell is an element where it is assumed that one dimension is much smaller compared to the other two directions. When the mesh is really fine, this assumption may no longer be valid and it could happen that the shell becomes too thick. When a shell gets too thick it can create negative Jacobians in the out-of-plane integration points, especially in areas of high curvature. To overcome this, you may need to remesh the affected area, adjust the shell thickness or even replace the shells with solid elements.

8.3.1.5 Rigid body modes

A rigid body mode is a mode of deformation that does not alter the stress in the body. Uniform translation or rotation of the entire model are two possible rigid body modes. In the presence of a rigid body mode, the solution is not uniquely defined and FEBio will most likely throw negative Jacobians. The solution is to identify the rigid body mode and to eliminate it by applying additional constraints to the model.

8.3.2 Failure to converge

FEBio uses an iterative method to solve the nonlinear finite element equations. This means that for each time step the solution for that step is obtained by a succession of iterations, where each iteration brings the model (hopefully) closer to the solution for that step. An iterative method

requires a convergence criterium to decide when to stop iterating and FEBio uses no less than three criteria. (In biphasic and multiphasic problems additional convergence criteria are used for the pressure and concentration degrees of freedom.) The convergence norm for each of these criteria is defined by the user in the control section of the input file.

Sometimes it happens that FEBio cannot converge on a time step. We'll take a look at some of the most common causes in the next following sections. Also take a look at possible causes of element inversions above. Sometimes, the issues that cause an element inversion may also manifest themselves in convergence problems before the element actually inverts.

8.3.2.1 No loads applied

Perhaps surprisingly, when no loads are applied, FEBio will often struggle to find a solution. The reason (and solution!) is simple. Although no loads are applied, FEBio still performs many calculations and due to round-off errors, the result of these calculations may render the convergence norms not exactly zero. FEBio then tries to apply the convergence criteria to these really small norms and again due to round-off error will not be able to satisfy the convergence criteria.

FEBio can actually detect this situation. It looks at the residual norm and when this is smaller than a user-defined limit it assumes that no loads are present and moves on to the next time step. However, this limit is actually problem dependent and it can happen that for your particular problem the limit is set too small. In that case, the solution is to increase the limit. This can be done by setting the *min_residual* parameter in the *Control* section of the input file. For example,

```
<min_residual>1e-15</min_residual>
```

If the residual norm now drops below this value, FEBio will consider the time step converged and move on to the next time step.

8.3.2.2 Convergence Tolerance Too Tight

In some cases, the default convergence tolerances are too tight and FEBio will not be able to converge. In that case, adjusting the affected convergence norms is a possible solution. This is done by editing the corresponding norms in the Control section of the input file. However, this should only be done when no other cause can be identified for the convergence problems.

8.3.2.3 Forcing convergence

It is possible to force a time step to converge. This can be done by interrupting the run (using ctrl+c) and then enter the *conv* command at the FEBio prompt. This will force the time step to converge and FEBio will continue with the next time step. This is not a recommended practice as the solution may become unstable and in fact it is unlikely that FEBio will be able to converge any of the following time steps. However, it can be useful in some scenarios. For example, if there is no load applied in the initial time step and FEBio has trouble getting past this initial step (see Section 8.3.2.1) or when you want to store the current state to the plot file without having to rerun the problem in debug mode (although the *plot* command or *debug* command might be better alternatives).

8.3.2.4 Problems due to Contact

When contact interfaces are defined, convergence problems can often be traced back to problems with the contact interfaces. See Section 8.5 for more details on how to properly use contact interfaces in FEBio.

8.4 Guidelines for Dynamic Analyses

A dynamic analysis differs from a quasi-static analysis in that the mass inertia of the model is taken into account. This adds an explicit time-dependency to the model (through the nodal velocities and accelerations) and therefore the solver needs to integrate the equations of motion in time. There are several integration schemes available in FEBio.

8.4.1 Implicit Dynamics Solver

The implicit solid solver implements the generalized alpha integration rule, which can be controlled via several parameters. Although the default settings will usually suffice, there might be instances where the parameters should be modified. (For example, see 8.4.3).

The main parameter that controls the time integration of the solid solver is the `rhoi` parameter. The following rules apply:

1. $\rho_\infty = -1$, All other integration parameters are set to one.
2. $0 \leqq \rho_\infty \leqq 1$, Newmark integration parameters are set to defaults, based on the value of ρ_∞.
3. Any other value for ρ_∞, default/user-specified values for Newmark integration parameters are not modified.

8.4.2 Explicit Dynamics Solver

The explicit solver uses a midpoint time integration rule.

An important aspect of the explicit solver is that is uses a “lumped” mass matrix, which reduces the mass matrix to a diagonal form. There are currently two lumping algorithms implemented:

row-sum The matrix elements of each row are summed together and the sum is assigned to the diagonal.

HRZ-lumping This is the preferred method of lumping for higher-order elements. However, it should not be used for shells.

8.4.3 False Transient Analysis

Quasi-static models may sometimes exhibit instabilities that may prevent the default Newton-based solver from converging. In such situations, a dynamic analysis can be tried instead, with the anticipation that the inertia of the model provides sufficient stability to solve the model successfully. Since in these situations, only the long-term quasi-static solution is sought, it is important to damp the oscillations introduced by the dynamics of the system. Such damping can be accomplished several ways.

The generalized-alpha time integration scheme of the implicit solver uses several parameters that control the overall energy-conservation of the model. By choosing these parameters such that the system looses energy over time, artifical damping can be introduced.

The `rhoi` control parameter controls the amount of artificial damping. This parameters takes a value between 0 (more numerical damping) and 1 (least numerical damping). Thus, by setting this value to 0, the most numerical damping can be introduced.

The `rhoi` control parameter sets the actual alpha, beta, gamma parameters of the generalized-alpha rule. Instead of using rhoi, users can set these actual parameters directly. To introduce the most numerical damping, set the values as follows.

```
<rhoi>-2</rhoi>
<gamma>1.5</gamma>
<beta>1</beta>
```

Note that rhoi needs to be set to -2 so that FEBio uses the actual alpha, beta, gamma parameters.

8.5 Guidelines for Contact Problems

FEBio offers many contact interfaces which allow the user to model complex boundary interactions. However, this capability comes at a price and the use of contact interfaces may create several problems in the solution process. It is hoped that the guidelines presented in this section may prevent many of the most common problems with contact.

Most of the contact algorithms implemented in FEBio offer two contact enforcement methods: a penalty method and an augmented Lagrangian method. Since the strategies for these two methods are slightly different, we will look at them separately. These algorithms are discussed in much more detail in the *FEBio Theory Manual*.

8.5.1 The penalty method

The penalty method enforces contact by "penalizing" (in an energy sense) any deviation from the contact constraint. In other words, when the two contacting surfaces penetrate, a force is generated proportional to the distance of penetration, which has the effect that the surfaces will now repell each other. The strength of this repelling force is controlled by the user through the *penalty factor*.

The obvious downside of this method is that some penetration is necessary to generate the contact force. The larger the penalty factor, the smaller this penetration needs to be and thus the better the contact constraint is enforced. However, choosing the penalty factor too large may cause the problem to become unstable since the contact force (and corresponding stiffness) will dominate the other forces in the model. Choosing a good penalty factor can thus be often difficult and may require several attempts before getting it "right". To help with choosing a penalty factor, FEBio offers the *auto_penalty* option for many of its contact interfaces which will estimate a good penalty factor based on element size and initial material stiffness. However, even then it may still take a few tries before getting the penalty factor good enough.

8.5.2 Augmented Lagrangian Method

In theory, in the presence of constraints, the corresponding Lagrange multipliers must be calculated which, in the case of contact, correspond to the contact forces that enforce the contact

constraint exactly. Unfortunately, the exact evaluation of these Lagrange multiplies is numerically very challenging and therefore in FEBio it was decided to evaluate these Lagrange multipliers in an iterative method, named the *Augmented Lagrangian* method. Using this method, FEBio will solve a time step several times (these iterations are termed *augmentations*), where each time the approximate Lagrange mutlipliers are updated (or *augmented*).

The obvious drawback of this method is that now each time step has to be solved several times. However, the advantage of avoiding the numerical problems of obtaining the exact Lagrange multipliers and the fact that in most contact problems very little extra work is needed to solve these augmentations, makes this method very attractive. In addition, it often gives better enforcement of the contact constraint compared to the penalty method.

So why not just use the augmented Lagrangian method? Well, often the penalty method will give good results and since the penalty method is much faster, it is often the preferred choice of many analysists.

In many cases, users are only interested in the final time step. For these users it may be of interest that it is possible to use the augmented Lagrange method only in the final time step. This can be done by defining a loadcurve for the *laugon* contact parameter. Then, define the corresponding loadcurve as zero everywhere except for the final time step. FEBio will now only use the augmented Lagrangian method in the final time step (and the penalty method all other steps).

8.5.3 Initial Separation

FEBio often struggles with problems that have an initial separation. This is especially so when the problem is force-driven, meaning that the deformation of the model is driven by a force (opposed to displacement-driven problems). In general, when the contact interface is necessary to define a well-constrained problem it is best to avoid initial separation if possible. Another way around this issue is to first use a displacement to bring the surfaces in contact and then replace the displacement with a force.

8.6 Cautionary Note for Steady-State Biphasic and Multiphasic Analyses

Biphasic, biphasic-solute and multiphasic analyses are generally transient analyses, involving mass transport (solvent and solutes) through a porous permeable domain. The default setting for these analyses is the transient mode (see Section 3.2). When a steady-state analysis is requested (*<analysis type="steady-state"/>*), the code simplifies the governing equations by setting time derivatives of the solid displacement and solute concentrations to zero. It is important to understand that this simplification is only applicable to analyses where interface conditions between sub-domains, and boundary conditions with the external environment, allow mass exchanges for the solvent and all solutes. If a domain or sub-domain is completely sealed such that it prevents solvent or solute exchanges, a steady-state analysis should not be used, as it would not capture these sealed conditions and would lead to erroneous results. For those types of problems, always use a transient analysis to obtain the correct solution. In those cases, a steady-state response may be obtained efficiently by temporarily increasing the transport properties (hydraulic permeability and solute diffusivities) by orders of magnitude via load curves, and/or by increasing the size of time steps.

8.7 Guidelines for Multiphasic Analyses

8.7.1 Initial State of Swelling

Under traction-free conditions, a multiphasic material is usually in a state of swelling due to the osmotic pressure difference between the interstitial fluid and the external environment. This osmotic pressure arises from the difference in interstitial versus external concentrations of cations and anions, caused by the charged solid matrix and the requirement to satisfy electroneutrality. An osmotic pressure difference arising from such electrostatic interactions is known as the *Donnan osmotic pressure*. When the Donnan pressure is non-zero, traction-free conditions do not produce stress-free conditions for the solid matrix, since the matrix must expand until its stressed state resists the swelling pressure.

The Donnan pressure reduces to zero when the fixed charged density is zero, or when the external environment is infinitely hypertonic (having ion concentrations infinitely greater than the interstitial fixed charge density). Since these two conditions represent special cases, it is generally necessary to devise methods for achieving the desired initial state of swelling, in an analysis where loads or displacements need to be prescribed over and above this swollen state. Swelling occurs as a result of the influx of solvent into the porous solid matrix. This influx is a time-dependent process that could require extensive analysis time. Therefore, it is computationally efficacious to achieve the initial state of swelling by using a multi-step analysis (Chapter 6) where the first step is a steady-state analysis (Section 3.3.1). In this steady-state step, the fixed charge density may be ramped up from zero to the desired value using a load curve for that property or, alternatively, the external environmental conditions may be reduced from a very high hypertonic state down to the desired level. In the second step, prescribed displacement boundary conditions (when needed) may be specified to be of type *relative* (Section 3.11.1), so that they become superposed over and above the initial swelling state.

Example:

```
<Module type="multiphasic"/>
<!-- rest of file -->
<Step>
  <Control>
    <analysis type="steady-state"/>
    ...
  </Control>
</Step>
<Step>
  <Control>
    ...
  </Control>
  <Boundary>
    <prescribe type="relative">
      <node id="22" bc="z" lc="4">1</node>
      ...
    </prescribe>
  </Boundary>
</Step>
```

8.7.2 Prescribed Boundary Conditions

In most analyses, it may be assumed that the ambient fluid pressure and electric potential in the external environment are zero, thus $p_* = 0$ and $\psi_* = 0$, where the subscripted asterisk is used to denote environmental conditions. Since the external environment does not include a solid matrix, the fixed charge density there is zero. For example, for a triphasic analysis, $c_*^+ = c_*^- \equiv c_*$. It follows that the effective fluid pressure in the external environment is $\tilde{p}_* = -R\theta\Phi_* \sum_\alpha c_*^\alpha$ and the effective concentrations are $\tilde{c}_*^\alpha = c_*^\alpha/\hat{\kappa}_*^\alpha \tilde{c}_*^\alpha$. Therefore, in multiphasic analyses, whenever the external environment contains solutes with concentrations c_*^α, the user must remember to prescribe non-zero boundary conditions for the effective solute concentrations *and* the effective fluid pressure.

Letting $p_* = 0$ also implies that prescribed mixture normal tractions (Section 3.13.2.5) represent only the traction above ambient conditions. Note that users are not obligated to assume that $p_* = 0$. However, if a non-zero value is assumed for the ambient pressure, then users must remember to incorporate this non-zero value whenever prescribing mixture normal tractions. Similarly, users are not required to assume that $\psi_* = 0$; when a non-zero value is assumed for the electric potential of the external environment, the prescribed boundary conditions for the effective concentrations should be evaluated using the corresponding partition coefficient, $\tilde{c}_*^\alpha = c_*^\alpha/\tilde{\kappa}_*^\alpha$.

8.7.3 Prescribed Initial Conditions

When a multiphasic material is initially exposed to a given external environment with effective pressure $\tilde{p}_*$ and effective concentrations $\tilde{c}_*^\alpha$, the initial conditions inside the material should be set to $\tilde{p} = \tilde{p}_*$ and $\tilde{c}^\alpha = \tilde{c}_*^\alpha$ in order to expedite the evaluation of the initial state of swelling. The values of $\tilde{p}_*$ and $\tilde{c}_*^\alpha$ should be evaluated as described in Section 8.7.2.

8.7.4 Prescribed Effective Solute Flux

The finite element formulation for multiphasic materials in FEBio requires that the natural boundary condition for solute α be prescribed as $\tilde{j}_n^\alpha \equiv j_n^\alpha + \sum_\beta z^\beta j_n^\beta$, where $\tilde{j}_n^\alpha$ is the effective solute flux. For a mixture containing only neutral solutes ($z^\beta = 0, \forall\beta$), it follows that $\tilde{j}_n^\alpha = j_n^\alpha$.

8.7.5 Prescribed Electric Current Density

The electric current density in a mixture is a linear superposition of the ion fluxes,

$$\mathbf{I}_e = F_c \sum_\alpha z^\alpha \mathbf{j}^\alpha \,.$$

Since only the normal component $j_n^\alpha = \mathbf{j}^\alpha \cdot \mathbf{n}$ of ion fluxes may be prescribed at a boundary, it follows that only the normal component $I_n = \mathbf{I}_e \cdot \mathbf{n}$ of the current density may be prescribed. To prescribe I_n, it is necessary to know the nature of the ion species in the mixture, and how the current is being applied. For example, if the ions in the triphasic mixture consist of Na^+ and Cl^-, and if the current is being applied using silver/silver chloride (Ag/AgCl) electrodes, a chemical reaction occurs at the anode where Ag combines with Cl^- to produce AgCl, donating an electron to the electrode to transport the current. At the cathode, the reverse process takes place. Therefore, in this system, there is only flux of Cl^- and no flux of Na^+ ($j_n^+ = 0$) at the electrode-mixture interface, so that the prescribed boundary condition should be $j_n^- = -I_n/F_c$. Since $z^+ = +1$ and $z^- = -1$ in a triphasic mixture, the corresponding effective fluxes are given by $\tilde{j}_n^+ = 2j_n^+ - j_n^- = I_n/F_c$ and $\tilde{j}_n^- = j_n^+ = 0$.

8.7.6 Electrical Grounding

If a multiphasic mixture containing ions is completely surrounded by ion-impermeant boundaries it is necessary to ground the mixture to prevent unconstrained fluctuations of the electric potential. Grounding is performed by prescribing the effective concentration of one or more ions, at a location inside the mixture or on one of its boundaries corresponding to the location of the grounding electrode. The choice of ion(s) to constrain may be guided by the type of grounding electrode and the reference electrolyte bath in which it is inserted.

8.8 Guidelines for Fluid Analyses

Currently, the fluid solvers of FEBio can be used to solve for isothermal fluid mechanics problems ("fluid" solver), isothermal fluid-structure interaction problems ("fluid-FSI" solver, Section 8.8.6) and isothermal fluid mechanics with solute (mass) transport ("fluid-solutes" solver). When solutes are included in an analysis, their treatment is very similar to the "multiphasic" solver described in Section 4.12. The fluid pressure p and solute concentrations c^ι in a fluid-solutes domain are thus related to the effective fluid pressure $\tilde{p}$ and effective solute concentrations $\tilde{c}^\iota$ as per eq.(4.12.7), to account for the possibility that the solutes are electrically charged. The theoretical treatment of solutes in the "fluid-solutes" solver follows that of the "multiphasic" solver, with the following simplifications: (a) The effective solubility $\hat{\kappa}^\iota$ appearing in the expression for the partition coefficient $\tilde{\kappa}^\iota$ in eq.(4.12.7) only embodies non-ideal physico-chemical behavior (whereas $\hat{\kappa}^\iota$ in a multiphasic mixture with a porous solid matrix can also embody the steric volume exclusion of solutes from the pore space); therefore, one should always set $\hat{\kappa}^\iota = 1$ when modeling ideal behavior in a "fluid-solutes" analysis. (b) Since there is no porous solid matrix that can hinder solute transport, the diffusivity tensor $\mathbf{d}^\iota$ of a solute in a "fluid-solutes" mixture is always equal to $d_0^\iota \mathbf{I}$, where d_0^ι is the solute diffusivity in free solution. Thus, even though FEBio requires users to specify a value for $\mathbf{d}^\iota$, that value is essentially ignored in the "fluid-solutes" solver code, where it is replaced by $d_0^\iota \mathbf{I}$. If users would like to neglect osmotic pressure in a "fluid-solutes" analysis, the osmotic coefficient Φ should be set to zero, producing $\tilde{p} = p$. Moreover, in the absence of solutes ($c^\iota = 0$ for all ι, as in the "fluid" and "fluid-FSI" solvers) this relation holds automatically. Therefore, when dealing with the "fluid" and "fluid-FSI" solvers, we do not need to add the "effective" modifier when mentioning the fluid pressure.

There are two broad categories of materials available for "fluid-FSI" analyses: (1) *fluid-FSI*" (Section 4.16.3), and (2) *biphasic-FSI* (Section 4.16.4). A *fluid-FSI* material is used to describe a fluid domain whose mesh is deformable; though a solid material (typically *neo-Hookean*) must be assigned to the mesh of a *fluid-FSI* material, its mass density is ignored and it is the user's responsibility to ascribe a negligible elasticity to this solid domain, whose sole purpose is to regularize the mesh deformation. A *biphasic-FSI* material considers that the deformable domain is a biphasic material (see Section 4.10), with a porous-permeable solid domain that has non-negligible mass density, solid volume fraction, solid matrix stiffness, and hydraulic permeability. The interstitial fluid of such a domain may also be ascribed a viscosity (Newtonian or non-Newtonian).

8.8.1 Degrees of Freedom and Boundary Conditions

The nodal degrees of freedom for "fluid" analyses are the nodal fluid velocity $\mathbf{v}^f$ and nodal fluid dilatation e^f. For a "fluid-FSI" analysis the nodal fluid velocity $\mathbf{v}^f$ is replaced with the relative fluid flux $\mathbf{w} = \varphi^f \left(\mathbf{v}^f - \mathbf{v}^s\right)$ and supplemented with the nodal solid displacement, $\mathbf{u}$, whose material

time derivative equals the solid velocity $\mathbf{v}^s$. Here, φ^f represents the fluid volume fraction (porosity) of the fluid-FSI domain, which is prescribed by the user in a *biphasic-FSI* material (via the complementary solid volume fraction) or automatically set to $\varphi^f = 1$ in a *fluid-FSI* material. Thus, the degrees of freedom of a fluid-FSI analysis reduce to those of a fluid (or fluid-solutes) analysis in the limit when $\varphi^f \to 1$ and $\mathbf{v}^s \to \mathbf{0}$. For "fluid-solutes" analyses, the nodal degrees of freedom include the effective solute concentrations $\tilde{c}^\iota$; as for "multiphasic" analyses, it is assumed that solutes occupy a negligible volume fraction in the mixture (thus $\varphi^f = 1$ for "fluid-solutes" analyses).

The fluid Cauchy stress is given by $\boldsymbol{\sigma}^f = -p\mathbf{I} + \boldsymbol{\tau}$, where p is the actual (not the effective) fluid pressure, as per eq.(4.12.7). The viscous stress $\boldsymbol{\tau}$ depends on the fluid rate of deformation tensor $\mathbf{D}^f$ according to a user-selected constitutive relation, such as *Newtonian fluid*.

In any of the fluid analysis types, the user may prescribe any of the components of $\mathbf{w}$ as an essential boundary condition, or the corresponding component of the viscous traction $\mathbf{t}^\tau = \boldsymbol{\tau} \cdot \mathbf{n}$ as a natural boundary condition ($\mathbf{n}$ being the normal to the boundary surface). When neither a component of $\mathbf{w}$ nor the corresponding component of $\mathbf{t}^\tau$ is prescribed explicitly, the latter is naturally assumed to be zero. Similarly, the user may prescribe e^f (or $\tilde{p}\left(e^f\right)$, see Section 3.13.2.13) as an essential boundary condition, or $w_n = \mathbf{w} \cdot \mathbf{n}$ as a natural boundary condition. When neither is prescribed explicitly, it is naturally assumed that $w_n = 0$. Natural boundary conditions are useful for creating symmetry planes in fluid analyses, where the normal fluid velocity and the viscous traction are zero: On those symmetry planes, no boundary conditions should be specified.

It is noteworthy that the fluid formulation in FEBio allows the prescription of the nodal value of $\mathbf{w}$ as an essential boundary condition, and the surface value of w_n as a natural boundary condition. On boundaries where the fluid velocity is completely prescribed, i.e., when normal and tangential components are known (and not zero), prescribing $\mathbf{w}$ and w_n instead of just $\mathbf{w}$ produces the best computational outcome. In those cases, the user should use the *fluid velocity* surface load, which combines both boundary conditions (Section 3.13.2.18). If the user only wants to prescribe a known normal velocity w_n and leave the tangential velocity unspecified, use the *fluid normal velocity* surface load (Section 3.13.2.17); this surface load also provides the option of setting the tangential fluid velocity to zero. Finally, if the user wants the velocity to remain normal to the selected surface (e.g., an inlet or outlet surface on which e^f is prescribed or fixed), but does not know the velocity magnitude *a priori*, use the *normal fluid velocity* constraint (Section 3.15.4) to enforce this condition.

A viscous fluid satisfies the no-slip condition on a physical boundary surface. This means that $\mathbf{w} = \mathbf{0}$ on no-slip boundaries. This boundary condition can be enforced simply by fixing the three components of $\mathbf{w}$ (Section 3.11.6) and not specifying any value for e^f on that surface (which naturally enforces $w_n = 0$).

Computational fluid dynamics analyses are often performed over a mesh that truncates the real fluid domain. Thus, fluid may enter the finite element domain across some upstream inlet boundary and leave the domain across some downstream outlet boundary. The exact flow conditions on these boundaries may not be known, thus they have to be approximated using best guesses. Arguably, the viscous traction $\mathbf{t}^\tau$ is the least intuitive boundary condition to guess. On an outlet boundary, it is necessary to prescribe or fix the dilatation e^f to overcome the natural boundary condition $w_n = 0$. However, in most cases the tangential components of $\mathbf{w}$ on that boundary are not known, therefore they cannot be fixed or prescribed, leading to the natural condition $\mathbf{t}^\tau = \mathbf{0}$. This natural condition may work fine in some cases, but it will often fail numerically when vortices being shed inside the finite element domain try to cross the outlet boundary. In those cases, consider using the *fluid backflow stabilization* (Section 3.13.2.15) and *fluid tangential stabilization* (Section 3.13.2.16) surface loads, which prescribe a velocity-dependent viscous traction $\mathbf{t}^\tau$.

Similarly, on an inlet boundary that shares an edge with a no-slip surface, prescribing the fluid

velocity $\mathbf{w}$ on the inlet boundary may not accurately reproduce the velocity profile in the boundary layer that would be produced on the adjoining no-slip surface. This potential mismatch could result in numerical instabilities; in such cases, prescribing a dilatation e^f on the shared edge may mitigate this issue.

8.8.2 Biased Meshes for Boundary Layers

In fluid analyses, boundary layers will form in the vicinity of no-slip boundaries, where the velocity magnitude varies rapidly with the distance from a no-slip surface. The thickness of boundary layers decreases with increasing Reynolds number. To capture these boundary layers accurately in a fluid finite element analysis, it is necessary to refine the mesh to produce thinner elements closer to the no-slip boundaries. This is done most conveniently by using mesh biasing tools, available in various meshing software programs, or post-hoc boundary-layer meshing tool that modify an existing mesh. Both of these options are available in FEBioStudio. Boundary layer meshing should always be used in fluid analyses in FEBio. Whereas other finite element codes employ numerical stabilization techniques that partially alleviate the need for biased meshes, FEBio does not employ such techniques. Therefore, using uniform finite element meshes may fail to produce numerical convergence, even in some seemingly simple problems. Users should keep in mind that biased meshes don't necessarily require more nodes and elements, therefore there is no automatic computational cost for using biased meshes.

8.8.3 Computational Efficiency: Broyden's Method

Fluid analyses produce a non-symmetric stiffness matrix and the resulting system of equations may be efficiently solved using Broyden's quasi-Newton method, where an approximation to the matrix inverse is produced at each iteration after the first full-Newton iteration of a time step. In fluid analyses, it is often possible to achieve convergence at each time step without having to perform additional full-Newton updates. Therefore, it is recommended to set *max_updates* (Section 3.3) to a large number (e.g., 50), along with setting *diverge_reform* to 0 (false), which leads to a more efficient solution scheme. A further advantage arises in fluid analyses since the mesh remains invariant over time: it is often possible to continue using Broyden updates even across time steps, without performing a full-Newton iteration at the start of that step, by setting *reform_each_time_step* to 0 (false). In that case, the solver will continue using Broyden updates up to the value of *max_updates*, before performing a full Newton update.

8.8.4 Dynamic versus Steady-State Analyses

FEBio runs fluid analyses in dynamic mode by default, because many fluid flow problems do not have a steady-state solution, since the governing equations are nonlinear. For example, many flows may exhibit vortex shedding as fluid flows across an obstacle, such as a flow constriction or a solid body. Even if some form of periodicity emerges under specific flow conditions, as in the von Karman vortex street, the corresponding solution is not in steady state. However, for some low Reynolds number laminar flows, a steady-state response may exist and FEBio will attempt to find that solution when using analysis type "steady-state" (Section 3.3). The user should be aware that, under steady-state analyses, the solution may not necessarily converge as efficiently to the final solution as would a dynamic analysis that allows the solution to reach steady state after a sufficiently long time. The best choice of analysis type for steady-state problems may need to be determined by trial and error.

8.8.5 Isothermal Compressible Flow versus Acoustics

The governing equations for fluid flow in FEBio represent isothermal conditions, thus eliminating temperature as a variable in the numerical analysis. Since the fluid solver accommodates some measure of fluid compressibility (via the fluid dilatation variable e^f), it is possible to solve for pressure wave propagation in fluids using this formulation. However, from a theoretical perspective, it should be recognized that the speed of sound (i.e., the wave speed) in isothermal (constant temperature) flow is different from the speed of sound in isentropic (constant entropy) flow. The field of acoustics typically examines pressure wave propagations under isentropic conditions, which are found to be more consistent with experimental measurements of the speed of sound in air. Therefore, the FEBio fluid solver may not produce realistic wave speeds when simulating acoustics. A fluid solver for isentropic flow may be developed for FEBio in the future.

8.8.6 Fluid-Structure Interactions

Fluid-structure interactions (FSI) may be modeled in FEBio using the *fluid-FSI* module (Section 3.2). In an FSI analysis the finite element mesh defining the fluid domain is deformable, so that the shape of the fluid domain does not have to remain fixed over time. The fluid flows through the deforming mesh, just as it does in standard computational fluid dynamics (CFD) analyses with fixed meshes. The deformation of the fluid domain may be controlled by prescribing suitable boundary conditions (so-called *moving boundary problems*) and by interactions with solid domains. For example, fluid may flow inside a flexible tube and the fluid pressure may cause the tube to expand or contract; the tube itself may be pinched at some location, reducing the cross-section through which fluid flows. The tube wall is a deformable solid domain which may be modeled in FEBio using any of the solid material models described in Sections 4.1 through 4.9, or it may be modeled as a porous deformable domain containing a viscous fluid, in which case one should use the *biphasic-FSI* material for the tube wall (Section 4.16.4). The inner tube (the fluid domain) should be modeled as a *fluid-FSI* material (Section 4.16.3).

In FEBio, a *fluid-FSI* material is modeled as a specialized fluid-solid mixture where the solid matrix density is zero and its stiffness should be set to a negligible value. In contrast, a *biphasic-FSI* material is a full-fledged fluid-solid mixture, where the solid may have mass and exhibit any desired nonlinear elastic or inelastic response. Both FSI materials require the user to define a *fluid* material from the same list as those available for CFD analyses (Section 4.16). It also requires the user to define a *solid* material to regularize the mesh deformation, which may be selected from Sections 4.1 through 4.9 for a *biphasic-FSI* material. In contrast, the *solid* material of a *fluid-FSI* material is massless (the *density* value entered by the user is ignored and reset internally to zero). Among all the choices available in Section 4.1, it is recommended to use the *neo-Hookean* elastic solid (Section 4.1.4.19), setting Poisson's ratio ν to zero and Young's modulus E to a very low value (a few orders of magnitude smaller than the moduli of surrounding solid domains).

The degrees of freedom at each node of a FSI domain consist of the components of the solid displacement $\mathbf{u}$ (which describe the mesh deformation), the fluid velocity $\mathbf{w}$ relative to the solid, and the fluid dilatation e^f. Essential boundary conditions may be prescribed on any of these degrees of freedom. For example, in a moving boundary problem, the motion of a boundary may be prescribed by setting the relevant components of $\mathbf{u}$. A no-slip boundary between an FSI material and a solid domain may be defined by simply setting $\mathbf{w} = \mathbf{0}$, regardless of the motion of that boundary. In general, all boundary conditions described for CFD analyses in Section 8.8.1 may be applied identically to FSI analyses.

Currently in FEBio, the mesh of a *fluid/biphasic-FSI* material domain must be continuous with

the mesh of surrounding solid domains. The surface between domains, which consists of the shared faces of adjoining finite elements from each domain, is called the FSI interface. Because the mesh is continuous, the solid displacement and the solid component of the traction are automatically continuous across FSI interfaces; when fluid is present on both sides of the interface, the relative fluid flux is also automatically continuous. In other words, when two FSI domains are adjoining (e.g., fluid-FSI with fluid-FSI, or fluid-FSI with biphasic-FSI), all boundary conditions are automatically satisfied across those interfaces. However, when an FSI domain interfaces with a solid domain, the fluid component of the traction in the *fluid-FSI* and *biphasic-FSI* domains must be explicitly transferred to the solid domain across that interface. This is done by prescribing a *fluid-FSI traction* or a *biphasic-FSI traction* on that interface (Section 3.13.2.20). This surface load does not require any user-defined parameters; its sole purpose is to identify those FSI interfaces where the fluid traction must be properly transferred to the surrounding solid domain. Omitting this boundary condition means that the fluid traction (i.e., the fluid pressure and the normal and tangential viscous components of the fluid stress) have no effect on this interface; the interface may still deform in response to the motion of the solid domain. For example, when a fluid interacts with a rigid solid domain whose motion is entirely prescribed, it is not necessary to apply a *fluid-FSI traction* at the interface between the fluid and rigid solid domains.

A *fluid-FSI traction* is also required on free fluid surfaces, such as the surface of a fluid in open channel flow, in order to properly capture the motion of that surface (such as waves). This condition is required since the net traction $\mathbf{t}$ on a free surface is zero. As the fluid-FSI domain is modeled as a specialized solid-fluid mixture, the net traction consists of the sum of solid and fluid tractions, $\mathbf{t} = \mathbf{t}^s + \mathbf{t}^f$. Setting $\mathbf{t}$ to zero produces a traction $\mathbf{t}^s = -\mathbf{t}^f$ on the solid mesh, allowing it to deform to the proper shape.

Here are a few cautionary notes to keep in mind when running FSI analyses: (1) Depending on the problem being analyzed, fluid-structure interactions may cause significant mesh deformations that lead to mesh distortion. Currently, adaptive meshing is not yet available to relieve this mesh distortion, therefore analyses and meshes need to be designed to minimize this effect or to terminate before the solution becomes too corrupted. (2) It is possible to pinch a solid domain that surrounds a fluid domain until the flow cross-section has reduced considerably; however it is not possible to completely shut off the flow, as this would require reducing the volume of some finite elements to zero. (3) While it is possible to model the deformation of free fluid surfaces, such as the formation of surface waves in open channel flow or the deformation of fluid blobs floating in a gravity-free environment, it is not possible to model the separation of such fluid domains into sub-domains such as spraying fluid droplets; this means that analyses where such phenomena are expected to happen will likely fail. (4) Whereas standard CFD analyses may be jump-started by prescribing a relatively large fluid velocity at the very first time step of an analysis, these types of boundary conditions may lead to dynamic oscillations and potential instabilities in an FSI analysis; therefore, users must be more considerate when prescribing boundary conditions for FSI analyses.

8.8.7 Fluid-Solutes Analyses

Fluid-solutes analyses introduce two types of molar fluxes into an analysis: (1) The diffusive molar flux $\mathbf{j}_d^\iota = -\tilde{\kappa}^\iota d_0^\iota \operatorname{grad} \tilde{c}^\iota$ and (2) the sedimentation flux $\mathbf{j}_b^\iota = s^\iota c^\iota \mathbf{b}^\iota$, where $s^\iota \equiv d_0^\iota \frac{M^\iota}{R\theta}$ is the sedimentation coefficient and $\mathbf{b}^\iota$ is the body force acting on solute ι. These molar fluxes represent the flux

of the solute relative to the solvent. The natural boundary condition for solutes is $\tilde{j}_n^\iota = 0$, where

$$\tilde{j}_n^\iota = \left(\mathbf{j}^\iota + \sum_\gamma z^\gamma \mathbf{j}^\gamma\right) \cdot \mathbf{n} \tag{8.8.1}$$

is the normal component of the effective molar flux $\mathbf{j}^\iota + \sum_\gamma z^\gamma \mathbf{j}^\gamma$ on a boundary, with $\mathbf{j}^\iota = \mathbf{j}_d^\iota + \mathbf{j}_b^\iota$ representing the sum of diffusive and sedimentation fluxes. The effective molar flux is constructed in this manner to enforce the electroneutrality condition,

$$\sum_\iota z^\iota c^\iota = 0 \tag{8.8.2}$$

as also done in "multiphasic" analyses (Section 4.12.1.4).

Zero natural flux $\tilde{j}_n^\iota$ implies that the solute moves at the velocity of the solvent (the fluid). Therefore, if the fluid satisfies $v_n^f = 0$ on that boundary, the solute will not transport across it. But if the fluid is allowed to transport across the boundary (e.g., using a prescribed fluid pressure), the solute can also transport across the boundary via convection. Therefore, for open flows or flows through conduits, where the fluid pressure is prescribed (e.g., set to zero) at an outlet boundary, no specific boundary condition needs to be prescribed on the solute in order for the latter to convect freely with the solvent at that outlet boundary. However, if the solvent exhibits back flow on that boundary in a manner that requires the application of the *fluid backflow stabilization* (Section 3.13.2.15) and *fluid tangential stabilization* (Section 3.13.2.16) surfaces loads, then consider also using the *solute backflow stabilization* surface load (Section 3.13.2.22) on that boundary, to produce more physically-realistic solute flows on such truncated domains, and improve numerical convergence.

The fluid pressure p in a fluid-solutes analysis is given by eq.(4.12.7). To prescribe the fluid pressure p on the boundary of a fluid-solutes domain, use the fluid pressure boundary condition specified in Section 3.11.16.

By default, the fluid-solutes solver employs the *staggered equation_scheme* format for the stiffness matrix, and solves for the nodal fluid velocity $\mathbf{v}^f$, nodal dilatation e^f and nodal effective solute concentrations $\tilde{c}^\iota$ using a *coupled solve_strategy*. This default settings works well when the Peclet number (the ratio of solute convective velocity to solute diffusive velocity) is relatively low (e.g., less than 100). For these problems, using the default value of *ctol=0.01* is generally appropriate. However, if users notice significant convergence difficulties on the solute concentration as the Peclet number approaches 100 or greater, consider setting *ctol=0*. When solving strongly convective solute transport problems with high Peclet numbers (e.g., greater than 1000), diffusion will be negligible compared to convection. In that case, users should switch to *equation_scheme=block* and *solve_strategy=sequential* and set *ctol=0* to increase the chances of achieving a converged numerical solution. The rationale for these choices is explained in the FEBio journal article associated with this fluid-solutes solver [70].

As done in the multiphasic solver, the solute volume fraction is considered negligible in fluid mixtures modeled in the fluid-solutes solver. Therefore, the volume of a fluid mixture will not change with increasing solute concentrations. When a solute is convected by the solvent, it means that the solvent exerts a frictional force on the solute, whose magnitude is inversely proportional to the solute diffusivity d_0^ι. In the fluid-solutes solver, the acceleration of solutes is considered negligible, implying that the velocity of the solute in the solvent reaches its terminal velocity instantaneously (thus, inertial forces are negligible compared to the drag force between solute and solvent).

According to Newton's third law stating that every action force has an equal and opposite reaction force, the drag force exerted by the solvent on the solute is balanced by an equal and

opposite reaction force exerted by the solute on the solvent. However, by default, this reaction force on the solvent is turned off in the fluid-solutes solver. In other words, the motion of the solvent affects that of the solute, but the motion of the solute does not affect the solvent. This default setting is adopted because it represent the common form of diffusion-reaction equations modeled in most of the mass transport literature. Therefore, when modeling solute diffusion problems in a stationary solvent (e.g., in the absence of pressure or velocity-driven solvent flows), the diffusive flux of a solute will not drag the solvent, keeping it in a stationary state. When the solvent motion is driven by solute concentration gradients, we call it *osmosis*. There is no common term describing solvent motion caused by solute sedimentation (motion of solute under the action of a body force), so we use the term *osmosis* to describe this phenomenon as well. Users can override the default setting of the fluid-solutes solver by checking the *include osmosis* box in the fluid-solutes material definition. When this box is checked, the solvent will have a force exerted on it by the solute (either due to diffusion or sedimentation, or both), causing it to move.

High-Peclet number flows are notoriously difficult to solve numerically. In FEBio we have reported valid solutions for a Peclet number as high as 10^{11} [70], thanks to the specific finite element formulation employed in the fluid-solutes solver, and the use of the sequential solver when needed. Nevertheless, the fluid-solutes solver is not fool-proof. In some problems, spurious increases in solute concentration can occur, seemingly randomly in the mesh, even when using a highly regular mesh. In many of these cases, this type of numerical artifact may go away by simply refining the mesh. However, users should be aware that this artifact may emerge 'silently', meaning that it does not necessarily produce slower numerical convergence that could hint at this problem.

The explanation for this behavior is that the numerical solutions to the effective solute concentrations at large Peclet numbers are effectively solutions to the partial differential equation $D^f\left(J^f c^\iota\right)/Dt = 0$, as reviewed in the Theory Manual. This integration is performed based on the current solution for $J^f = 1 + e^f$ and $\mathbf{v}^f$ as well as the structure of the mesh. Any ill-conditioning in the mesh (such as a mesh which is too coarse) may introduce a numerical round-off error in the calculation of $\tilde{c}^\iota$, and there is no other sufficiently dominant term in the solute mass balance equation that can dampen or correct for such numerical round-offs. Thus, they may grow over time. Mesh refinements are required when encountering these circumstances.

Other boundary conditions for the solute (such as the essential boundary condition $\tilde{c}^\iota = 0$) may also improve predictions when using relatively coarse meshes, preventing the occurrence of spurious rises which often initiate on boundaries. Unfortunately, in some problems, this option forces users to adopt a non-physical boundary condition which may produce unacceptable results. This workaround may not be needed when using a sufficiently refined mesh, though finer meshes typically require longer solution times. Also note that highly irregular meshes near no-slip boundaries may cause spurious rises in solute concentration in high Peclet number flows. Therefore, it is recommended to use suitable meshing techniques for generating boundary layer meshes, such as stacking pentahedral elements atop tetrahedral elements when using unstructured tetrahedral meshes. For diffusion-dominated or reaction-dominated flows however, these numerical concerns may not arise. Nevertheless, a thorough understanding of the underlying physics of these problems is essential for properly prescribing boundary conditions, and the number and size of time increments in a finite element analysis.

The default settings for the fluid-solutes solver force it to reform the stiffness matrix at the start of each time step. This is a standard requirement in most of the FEBio solvers (including solid and structural mechanics, and standard biphasic and multiphasic domains), though it had previously been demonstrated that the fluid solver could avoid reforming the stiffness matrix even after achieving convergence at a given time step, producing a very efficient computational scheme. Unfortunately, this increased efficiency is not always an option with the fluid-solutes solver, because

the stiffness matrix contains terms that depend significantly on the solute concentrations. As these concentrations evolve over time and space, it becomes essential to update the stiffness matrix to reflect those changes. Therefore, consider using the direct sparse solver from the Intel MKL library (*linear_solver=Mkl dss*) to improve computational efficiency during matrix factorization, in comparison to the Pardiso solver from the same library. Moreover, in applications where the initial solute concentration is non-zero, and non-negligible compared to the prescribed change in concentration on an inlet boundary, it may be possible to solve a fluid-solutes problem more efficiently without reforming the stiffness matrix at each time step.

8.9 Understanding the Solution

Okay, FEBio found a solution. Great, but how do you know it is the right one? Well, it turns out this is in fact a harder question than it may seem. In any finite element model there are numerous assumptions and approximations and trying to discuss all of these is outside the scope of this section. Instead, we'll look at some of the most typical problems that FEBio users have run into.

8.9.1 Mesh convergence

The solution calculated by FEBio only applies to the mesh for which it was solved. However, the "exact" solution must be independent of the mesh and therefore a mesh convergence study is almost always necessary to make sure the FEBio solution is close to this exact solution. In practice this means that a model must be run several times with an increasingly finer mesh to find out at which mesh density the FEBio solution no longer changes.

8.9.2 Constraint enforcement

FEBio uses many iterative algorithms for enforcing constraints. For each of these contraints the user must verify that the solution indeed satisfies these constraints sufficiently. The two most common constraints used in FEBio are incompressibility and contact.

For uncoupled materials, incompressibility (for hexahedral elements) is handled in FEBio using a three-field formulation in addition to the enforcement of the incompressibility constraint. This constraint is usually enforced using a penalty formulation (although an augmented Lagrangian method is also available). To inspect whether the incompressibility constraint is satisfied sufficiently the user can look at the volume ratio which has to be close to one. If it deviates from one too much the user needs to increase the "bulk modulus" of the material.

For coupled materials, incompressibility is not treated explicitly and care must be taken when using coupled materials in a near-incompressible regime (e.g. setting the Poisson's ratio too close to 0.5 for a neo-Hookean material). In that case, the solution will most likely "lock" which manifests itself in displacements that are too small. Inspection of the volume ratio may not be sufficient to identify locking. However, a mesh convergence study will often help in identifying this problem.

When a contact constraint is not sufficiently enforced, the contacting surfaces will have penetrated. This problem is usually identified easily by looking at the deformed model in a post-processing software (e.g. PostView). Increasing the penalty factor and/or decreasing the augmented Lagrangian tolerance should solve this problem.

8.10 Guidelines for Using Prestrain

There are a variety of ways in which prestrain can be added to an FEBio model, but the recommended approach – a two-step approach – is outlined here.

In the first step, the prestrain is enforced on the model. This is accomplished by making sure the part that will be prestrained uses the prestrain elastic material (or uncoupled prestrain elastic for uncoupled hyperelastic materials). A suitable prestrain generator must be defined as well. In addition, one of the update rules can be specified as a nonlinear constraint in order to modify the effective prestrain field. The boundary conditions in this step should be minimal but sufficient to ensure a well-defined finite element problem.

After the first step converges, the model is successfully prestrained. However, depending on the application, the reference geometry could have changed and this may complicate the application of further loads (most of FEBio will still use the original geometry as the reference geometry). To eliminate this potential source of confusion, it is recommended to apply the prestrain initial condition (see chapter 4) which essentially fixes the prestrain gradient and resets the reference geometry to the current geometry. FEBio will then apply any subsequent loads to this new reference geometry. One caveat of this approach is that the plotfile still stores the original reference geometry. To prevent inconsistencies in the plotfile it is recommend suppressing the creation of the plot file in the first step. This is done by setting the following control flag in the Control section of the first step.

```
<plot_level>PLOT_NEVER</plot_level>
```

Then, in the Control section of the second step, restore the plot level value.

```
<plot_level>PLOT_DEFAULT</plot_level>
```

Now, the new reference configuration resulting after the prestrain analysis step, will be the reference configuration that is used in the plot file.

8.11 Guidelines for using the CG-solid solver

The nonlinear conjugate gradient solver, CG-solid, is an alternative solution algorithm for static solid mechanics problems. It does not currently support dynamic problems, fluids, etc. This solver has advantages in several situations:

- It uses much less memory and therefore allows bigger problems to be solved on a given computer than the standard Newton-based solvers
- Its performance scales better for large problems and it may be faster for very large problems (but see the limitations below);
- For nonlinear problems with buckling or other instabilities it can converge much more reliably than the standard solver;
- For many problems it will converge in a single timestep.

However, it also has some disadvantages:

- For small models it is typically slower than the BFGS solver;

- Although it usually converges in a single timestep, it requires a much larger number of iterations to do so. It is therefore inefficient if many timesteps are needed to produce intermediate results or follow the progression of a problem;
- The number of iterations it requires to converge is proportional to the number of elements across the longest dimension of the mesh. It therefore performs best for compact problems with similar numbers of elements in each direction. For example a cube with equal numbers of elements in each direction would solve much faster than a flat sheet or a long, thin beam with the same total number of elements;
- It does not currently work well for shell elements or freely moving rigid bodies. The way it works is quite different from the Newton-based solver and the problem may need to be defined differently for optimum results:
- The convergence criteria in FEBio measure the change in the solution on each iteration as a fraction of the total change for the timestep. Because the CG solver converges in a much larger number of smaller steps, the convergence criteria must be set smaller than for the BFGS solver. The default values are dtol=1e-6 and etol=0.001, but it may be necessary to reduce these further to ensure full convergence. Check the solution carefully to make sure that it is correct and reduce dtol further if necessary.
- The rate of convergence is quite slow near to the solution so if the convergence tolerance is set too small the solution may take much longer.
- It is usually best to use a single timestep if possible, rather than many smaller steps as is normal for the Newton solver. More timesteps will greatly increase the solution time.
- Unlike the Newton-based solvers, prescribed displacements are applied only to the nodes they affect at first and take many iterations to propagate through the adjoining mesh. This means that they can often cause excessive deformation of the adjacent elements, for example a compressive prescribed displacement can push the surface layer of nodes right through the adjoining elements causing a negative Jacobian error. Unlike the BFGS solver, where prescribed displacements often give more stable convergence, it is better to apply forces where possible and to avoid prescribed displacements.
- If prescribed displacements are essential, it may be necessary to use smaller timesteps to avoid excessive distortion of the adjoining elements. In this case it is possible to slacken the convergence criteria for the intermediate steps and then apply a tighter tolerance for the final step to generate a correct solution.
- The preconditioner improves convergence when the stiffness of the nodes varies, for example because of different element sizes, different materials or features such as quadratic elements with midside nodes. For problems where all the elements are the same it offers no great advantage and may even cause slower convergence in some cases. It is therefore best to use it for irregular meshes, multiple materials, quadratic elements and contact problems, but for problems with regular meshes where all the elements are a similar size it may be better to turn it off.
- For problems with large deformation and displacements the solver may take a very large number of iterations to converge. The solver proceeds by many small, cautious steps, unlike the Newton-based solvers, which attempt to jump straight to the final answer on each

iteration. It is therefore much more reliable but may need thousands or tens of thousands of iterations to reach the solution.

8.12 Limitations of FEBio

If you run into a problem that you can't seem to solve, maybe you ran into a limitation of FEBio. This section discusses some important limitations of the software and how they may affect the outcome of your analysis.

8.12.1 Geometrical instabilities

A geometrical instability is a point in the solution at which several paths for continuing the solution are possible. Typical examples are buckling of a column. FEBio's Newton based solvers cannot handle buckling since usually at these points, the material's elasticity tangent is not uniquely defined. Although in some cases FEBio will be able to pass a buckling point, in most cases, FEBio will not be able to converge to a solution.

8.12.2 Material instabilities

A material instability is a point in the stress-strain response where the slope of the stress-strain curve effectively becomes zero. Since FEBio's Newton based solvers use this slope to advance the solution, a zero slope would mean an infinite step and FEBio will not be able to continue.

The slope (or more accurately the elasticity tangent tensor) does not have to be zero to cause problems. For materials that soften at large deformations, this slope may become so small that it renders the global stiffness matrix ill-conditioned. It is unlikely that FEBio will be able to continue and even if it finds a solution, it may not be useful.

8.12.3 Remeshing

FEBio is designed to solve problems that can undergo large deformations. However, when the deformations become too large it can happen that the finite element mesh cannot be distorted any further without inverting elements. In that case, FEBio will not be able to continue the solution and will most likely terminate with a negative jacobian error message.

One possible solution to this problem is to remesh the model at the point at which the deformation become too large, but currently FEBio does not have any remeshing capabilities. The only remedy at this point is to plan ahead and design your mesh in such a way that the elements will not invert in the range of deformation that you are interested in (e.g. place a butterfly mesh in sharp corners or make the mesh finer in areas of larger deformation).

8.12.4 Force-driven Problems

When the primary method of deforming the model is a force, the problem is said to be force-driven. (Opposed to a displacement-driven model where the deformation is caused by displacement boundary conditions.) Force-driven forces are inherently unstable and FEBio has limited capabilities of handling such problems[1]. The cause of this instability can easily be explained using

[1] As of FEBio 2.0, some features are available that may help with this issue, but they are still experimental.

a simple example. Imagine a box that is constrained in all directions except one. In the unconstrained direction, apply two equal but opposite forces on opposite faces. In order to prevent the box from flying away, the box must generate stresses which balance the applied forces exactly. Unfortunately, the numerical solution will only be approximate due to numerical round-off errors inherent in the calculation. Even the slightest deviation from balancing the forces exactly will create a net-force which in turn causes the model to fly off in the unconstrained direction (in which a rigid body mode now exists).

Usually, these problems are circumvented by adjusting the boundary conditions so that rigid body modes cannot exist. In the previous example this can be done by only modeling half of the box and enforcing a symmetry boundary condition. However, in some cases this is not possible. This is often the case in force-driven contact problems, where the contact interface is necessary to define a well-constrained model. When there is an initial separation (or even when the surfaces have no initial overlap) the initial contact force is zero, which is equivalent to having no contact enforcement, and thus the model is under-constrained. Therefore, for force-driven contact problems it is important to have some initial contact before starting the analysis.

8.12.5 Solutions obtained on Multi-processor Machines

Although technically not a limitation of FEBio it is important that users are aware that in some cases you may get a different convergence history or sometimes even a slightly different answer when you run the same model twice on a multi-processor machine. This is caused by the fact that on multi-processor machines the same calculations can be executed in a different order and due to the accumulation of numerical round-off errors may result in slightly different answers. The type of problems that are mostly affected by this are problems that are close too being ill-conditioned. Also, problems that have many time steps or require many iterations for each time step may be affected by this.

If you experience different answers for the same problem, try running the model on just one processor (if possible). See Section 2.6 for more information on how to run FEBio on multi-processor machines.

8.13 Where to Get More Help

When you get here, you may be ready to pull all your hair out, but fret not, all is not lost. In fact, some people may have run into a similar issue and found a solution. The first place to find these people is on the FEBio user's forum (https://forums.febio.org/). This forum contains hundreds of posts by FEBio users and is monitored by the FEBio developers who will always be more than happy to help you with your problems. In addition, this forum can be used to report bugs or to request a new feature. It is the ideal portal to find help with your problems so don't hesitate to use it!

Chapter 9

Configuration File

9.1 Overview

FEBio requires a configuration file to run. The purpose of this file is to store platform-specific settings such as the default linear solver. See Section 2.5 for more information on how to use this file. This section details the format of this file.

The configuration file uses an xml format. The root element must be *febio_config*. The required attribute *version* specifies the version number of the format. Currently this value must be set to "3.0". The following elements are defined.

Parameter	Description
default_linear_solver	Set the default linear solver for the platform (1)
import	Load a plugin file (2)
set	define an alias (3)
if_debug	Process child tags only for debug versions of FEBio (4)
if_release	Process child tags only for release versions of FEBio (4)
output_negative_jacobian	Output a detailed list of inverted elements when negative jacobians are encounte
print_model_params	Print the values of load controlled parameters at the start of each time step (6)
show_warnings_and_errors	Print warnings and errors to the screen (7)

Comments:

1. FEBio supports several linear solvers, such as Pardiso and Skyline. Not all solvers are available for all platforms. Only the Skyline solver is available for all platforms. As of version 1.2, the Pardiso solver is available on all platforms for which FEBio is distributed. See section 9.2 on how to configure the linear solvers.

2. As of FEBio 2.0 the user can create and use plugins designed for FEBio. These plugins extend the standard capabilities without the need to recompile the FEBio code. See Chapter 10 for more information on using plugins in FEBio.

3. As of FEBio 2.5 the configuration file supports aliases which can be used to define plugin paths. The *set* tag defines an alias. The name of the alias is specified via the *name* attribute. Aliases are accessed using the $(*name*) syntax. See below for an example.

4. The *if_debug/if_release* tags allow users to customize settings based on whether a debug or release version of FEBio is processing the configuration file. (Note that the debug version refers to how FEBio was compiled. It does not refer to the -g command line option.) The versions of FEBio distributed through the website are release versions. A debug version of FEBio can only be obtained by building FEBio with the _DEBUG preprocessor command.

5. When elements become inverted, FEBio prints an error message that "negative jacobians" are encountered. If you wish to see a detailed list of the inverted elements, you can set the *output_negative_jacobian* flag to 1.

6. The *print_model_params* parameter can be used to print values of load-controlled parameters to the screen and log file. If disabled, the values are not printed to the screen or log file. For models that use many load-controlled parameters this can be useful to reduce the amount of output generated by FEBio. By default, this parameter is on.

7. The *show_warnings_and_errors* parameter can be used to decide whether or not to print warning and error messages. The default value is on.

An example configuration file that sets the default linear solver to pardiso, defines an alias, and loads a plugin.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<febio_config version="3.0">
  <default_linear_solver type="pardiso"/>
  <set name="PluginsDir">C:\path\to\plugins</set>
  <import>$(PluginsDir)\myplugin.dll</import>
</febio_config>
```

9.2 Configuring Linear Solvers

FEBio offers several direct and iterative linear solvers that can be used to solve the linear system of equations of the FE model. Users can configure these linear solvers in the configuration file with the `default_linear_solver` tag. The particular linear solver is selected via the `type` attribute. The following table lists the possible values, the solver type, and the supported matrix format.

solver	description	type	matrix format
pardiso	A sparse direct solver, from the MKL library. (default)	D	S/U
skyline	A sparse direct solver, but not as efficient as pardiso.	D	S
mkl_dss	The DSS solver from the MKL library.	D	S/U
fgmres	An implementation of the GMRES algorithm, from MKL .	It	U
cg	An implementation of the CG algorithm, from MKL .	It	S
boomeramg	An algebraic multigrid solver, from the HYPRE library.	It	S/U
schur	A block based iterative solver.	It	B

D=direct solver, It=iterative solver, S = symmetric format, U = unsymmetric format, B = block format.

Iterative solvers can be configured with preconditioners, which attempt to reduce the condition number of the matrix. Preconditioning is highly recommended and often necessary for reducing the number of iterations needed by the iterative solver to converge the solution. Any of the linear solvers listed above can be used as a preconditioner. In addition, the following specialized preconditioners are also available.

preconditioner	description	matrix format
ilu0	Incomplete LU decomposition with no fill-in	unsymmetric
ilut	Incomplete LU decomposition with custom fill-in	unsymmetric
ichol	Incompletly Cholesky decomposition with no fill-in	symmetric
diagonal	Diagonal preconditioner made from the diagonal of the matrix	any

The following sections describe each of these solvers and preconditioners in more detail.

9.2.1 Pardiso

The Pardiso solver is an effecient sparse direct linear solver and is the default linear solver. FEBio uses the implementation from the MKL library. It does not require any configuration parameters. It can take symmetric, unsymmetric, and structurally symmetric sparse matrix formats.

9.2.2 Skyline

The Skyline solver was initially added for users who build their own copy of FEBio and did not have access to third party linear solvers. It is a sparse direct solver, but not as efficient as Pardiso and not recommended for large problems. When using this solver, it is highly recommended to set the `optimize_bw` flag in the model file's Control section, which attempts to minimize the matrix bandwidth. It does not require any configuration parameters and only works with symmetric matrices.

9.2.3 MKL_DSS

The MKL library also offers another direct solver, called DSS (Direct-Sparse-Solver). This solver is very promising and appears to perform better for many problems than the pardiso solver. Currently, in FEBio4, pardiso remains the default mostly for backward compatibility reasons, but it is often worthwhile to see if the mkl_dss solver performs better.

9.2.4 FGMRES

This iterative linear solver, from the MKL library, implements the GMRES algorithm, which is an efficient Krylov-based iterative solution strategy. It requires several configuration parameters, listed below, and only works with unsymmetric matrices. In the table below, **N** refers to the number of equations.

parameter	description	default
max_iter	The maximum number of iterations	min(N,150)
tol	relative residual tolerance	1e-6
abs_tol	absolute residual tolerance	1e-12
print_level	The amount of information printed (0,1, or 2)	0
fail_max_iters	Indicates whether reaching max iters is a failure or not (1)	1 (true)
pc_left	The left preconditioner	(none)
pc_right	The right preconditioner	(none)

Comments:

1. When using FGMRES as a preconditioner it is recommended to set the fail_max_iters flag to zero.

9.2.5 CG

This iterative linear solver, from the MKL library, implements the Conjugate Gradient algorithm, which is an efficient Krylov-based iterative solution strategy for symmetric systems. It requires several configuration parameters, listed below, and only works with symmetric matrices. In the table below, **N** refers to the number of equations.

parameter	description	default
max_iter	The maximum number of iterations	min(N,150)
tol	relative residual tolerance	1e-6
print_level	The amount of information printed (0,1, or 2)	0
pc_left	The left preconditioner	(none)

9.2.6 BoomerAMG

This solver, from the HYPRE library, is an adaptive multigrid solver. It often works better as a preconditioner to an iterative solver than as a stand-alone solver.

parameter	description	default
max_iter	The maximum number of iterations	20
tol	relative residual tolerance	1e-7
print_level	The amount of information printed (0,1, or 2)	0
max_levels	The max number of multigrid levels	25
coarsen_type	Method of grid coarsening	(library default)
use_num_funcs	Use dimensionality of solution variable in coarsening	0 (false)
relax_type	relaxation type	3
interp_type	interpolation type	6
strong_threshold	strong threshold	0.5
p_max_elmts	P max elements	4
num_sweeps	Number of sweeps	1
agg_num_levels	Aggressive number of levels	0
nodal	Nodal option	0 (false)
do_jacobi	Do jacobi preconditioning	0 (false)
fail_max_iters	Fail on max iterations (1)	1 (true)

Comments:

1. When using this solver as a preconditioner, it is recommended to set the fail_max_iters flag to 0.

9.2.7 Schur

The Schur solver is a linear solver that takes a 2x2 block structured matrix and solves the linear system by decomposing it in its Schur-complement form. Thus, instead of solving the following system directly,

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}$$

it first solves for v,

$$Sv = b - CA^{-1}a$$

where $S = D - CA^{-1}B$ is the Schur complement of A. Then the Schur solver solves for u,

$$u = A^{-1}(a - Bv)$$

In order to use this solver, users need to select how they wish to solve the A-block, and how to solve the Schur complement. For the A-block users can choose any of the linear solvers listed above, either direct or iterative. For the Schur complement users can only choose a Krylov-based iterative solver (FGMRES or CG) because FEBio does not form the Schur complement explicitly.

This solver is often more effective as a preconditioner for FGMRES. In this case, the A-block and Schur complement only need to be solved approximately. See the examples section below for some example configurations. This solver requires a block-structured matrix. In order to generate the block structure you need to set the `equation_scheme` control parameter to 1 in the model input file.

parameter	description	default
print_level	The amount of information printed (0,1, or 2)	0
schur_block	Take the Schur complement of A (=0) or D (=1)	0
do_jacobi	Do Jacobi preconditioning	0
zero_D_block	Is the D-block zero?	0 (false)
A_solver	The solver for the A-block (outside the Schur complement)	user specified
schur_solver	The iterative solver for solving Schur complement	user specified
schur_A_solver	The solver for the A block in the Schur complement	(same as A_solver)

9.2.8 Examples

Example 1. The default pardiso solver is defined in the configuration file using the following syntax. It does not require any parameters.

```
<default_linear_solver type="pardiso"/>
```

Example 2. This example shows how to set up the FGMRES solver with an ILU0 preconditioner.

```
<default_linear_solver type="fgmres">
  <max_iter>100</max_iter>
  <tol>1e-5</tol>
  <pc_left type="ilu0"/>
</default_linear_solver>
```

Example 3. This example sets up an FGMRES solver with a Schur preconditioner. For the Schur system we use pardiso for solving the A-block outside of the Schur complement and approximate the A-block inside the Schur complement with its diagonal. This solver requires a block-structured matrix. In order to generate the block structure you need to set the `equation_scheme` control parameter to 1 in the model input file.

```
<default_linear_solver type="fgmres">
 <print_level>2</print_level>
 <max_iter>100</max_iter>
 <tol>1e-5</tol>
 <abs_tol>1e-9</abs_tol>
 <pc_left type="schur">
   <print_level>0</print_level>
   <do_jacobi>0</do_jacobi>
   <A_solver type="pardiso"/>
   <schur_pc>0</schur_pc>
   <schur_A_solver type="diagonal"/>
   <schur_solver type="fgmres">
     <print_level>0</print_level>
     <max_iter>100</max_iter>
     <tol>0.05</tol>
     <fail_max_iters>0</fail_max_iters>
```

```
    </schur_solver>
   </pc_left>
  </default_linear_solver>
```

Chapter 10

FEBio Plugins

As of version 2.0 FEBio supports plugins. Plugins are dynamic libraries that extend the capabilities of FEBio at runtime without the need to recompile the entire source code. This offers the user a powerful mechanism for extending the default feature set of FEBio with little effort. In addition, the development of the plugin is independent of the FEBio source code, which implies that upgrading to a newer version of FEBio will not require a recompilation of the plugin[1]. Using plugins, users can create custom materials, boundary conditions, loads, output variables, etc., and even couple FEBio to other codes.

The *FEBio Developer's Manual* explains in detail how to create these plugins. In this section we outline how to use the plugins with FEBio.

10.1 Using Plugins

A plugin is compiled into a dynamic library (dll or dylib) or shared object (so) and contains the compiled code of the library. In order to use this plugin, you must add the path to the plugin file in the FEBio configuration file (see Chapter 9 for a discussion of the configuration file). For each plugin, an *import* item has to be added in this configuration file. For example,

```
<import>myplugin.dll</import>
```

You may need to add the full path to the plugin if FEBio has problems locating the plugin.

```
<import>C:\path\to\my\plugins\myplugin.dll</import>
```

When FEBio starts, it will first load all the plugins that are defined in the configuration file before it reads the input file. This way, sections in the input file that require the plugin's capabilities will already be available for use.

An alternative way for loading a plugin is using the **-import** command line option.

```
>febio -i input.feb -import myplugin.dll
```

In this example, FEBio will look for the file *myplugin.dll* in the current working directory. If the plugin file is located in a different folder than the current working directory, you must prepend the relative or absolute path to the file. Note that if the path contains spaces, you need to wrap it in quotes.

[1] When upgrading to a newer version of FEBio, it may be necessary to recompile the plugin if the plugin interface was changed. As long as the plugin remains compatible with the new interface, no code changes are required.

```
>febio -i input.feb -import "C:\path\to\my custom plugin\myplugin.dll"
```

10.2 Error Messages

If FEBio cannot find a plugin it will print an error message. There are several reasons why a plugin fails to load. Below is a list of possible error messages and some actions that can be taken to prevent the error.

- *"Failed to load the file"*: Most likely FEBio cannot find the plugin file specified in the configuration file. Make sure that the file name including the path is correct and that there are no spaces at the start or end of the import tag's value string.

- *"Required plugin function PluginNumClasses not found"*: The plugin is missing a required function that FEBio needs to communicate with the plugin. Notify the developer of the plugin to fix this.

- *"Required plugin function PluginGetFactory not found"*: The plugin is missing a required function that FEBio needs to communicate with the plugin. Notify the developer of the plugin to fix this.

- *"Invalid number of classes returned by PluginNumClasses":* There was a problem with the plugin's innitialization. Notify the developer of the plugin to fix this.

- *"Required plugin function GetSDKVersion not found"*: The plugin is missing a required function that FEBio needs to communicate with the plugin. Notify the developer of the plugin to fix this.

- *"Invalid SDK version"*: The plugin was compiled with a version of the SDK that is not compatible with the current FEBio version. This most likely means that the plugin must be rebuilt with a newer version of the SDK.

Appendix A

Heterogeneous model parameters

Many model parameters can be made dependent on the reference coordinates. This is useful, for instance, for defining inhomogeneous materials, where some material parameters depend on the spatial position.

There are two mechanisms for creating heterogeneous parameters: a mathematical expression, or a map. The type of the parameter is set via the *type* attribute, which can take on two values.

type	description
math	Define a math parameter via a mathematical expression
map	Define a mapped parameter
const	Define a constant parameter (default)

The two types of parameters are described in more detail in the sections below.

The *type* attribute can be omitted. In that case, it is assumed to be a const parameter, unless the tag's value is not a number. In that case, it is assumed to be a math parameter.

FEBio performs parameter validation at the start of an analysis where the values of parameters are checked against the valid ranges as defined in the code. However, it is important to keep in mind that this validation is only performed on constant parameters. For math and map parameters, it is up to the user to ensure that all values over the relevant mesh partition are valid.

A.1 Math parameters

For math parameters, add the type="math" attribute to the parameter's definition. The value of the parameter tag can then be any mathematical expression. The upper case letters X, Y, Z, are used to denote material coordinates. The letter *t* denotes time. The algebraic operators +, -, *, /, ^ are supported, as well as the list of functions listed in Appendix C.

```
<material id="1" name="my_material" type="neo-Hookean">
  <E type="math">X+Y+Z</E>
  <v>0.3</v>
</material>
```

You can also use the component's other parameters inside mathematical expressions.

```
<material id="1" name="my_material" type="neo-Hookean">
  <density>1.0</density>
  <E type="math">2*density+0.1</E>
  <v>0.3</v>
</material>
```

The names of maps are also allowed in mathematical expressions. For instance, assume that a map, named “E_map” is defined in the MeshData section. Then the following is a valid mathematical expression.

```
<E type="math">2*E_map+X</E>
```

A.2 Mapped parameters

For mapped parameters, use the attribute type=“map”. In this case, the value of the parameter is the name of the map that is defined in the MeshData section.

```
<material id="1" name="my_material" type="neo-Hookean">
  <E type="map">map_E</E>
  <v>0.3</v>
</material>
```

The map is defined in the MeshData section.

```
<ElementData name="map_E" elem_set="Part1">
  <elem lid="1">1.23</elem>
  <elem lid="2">4.56</elem>
</ElementData>
```

Note that it is also possible to use maps in mathematical expressions.

```
<material id="1" name="my_material" type="neo-Hookean">
  <E type="math">5.0*map_E</E>
  <v>0.3</v>
</material>
```

Appendix B

Referencing Parameters

This appendix details how that model parameters can be referenced in the FEBio input files. This is used to

- write model parameters to the output file.
- reference model parameters in the parameter optimization input file.

The general format follows the following rule:

```
fem.property.property...property.parameter.component
```

Properties and parameters are referenced by tags separated by a periods. The first tag must be the *fem* tag, which serves as the name of the model. (In future version this name can be user defined, but for now is fixed). Next, the name of a model property must follow. The following properties are currently defined:

- **material**: references a material, defined in the *Material* section.
- **bc_fixed**: references a fixed bc, defined in the *Boundary* section.
- **bc_prescribed**: references a prescribed bc, defined in the *Boundary* section.
- **nodal_load**: references a nodal load, defined in the *Loads* section.
- **surface_load**: references a surface load, defined in the *Loads* section.
- **body_load**: references a body load, define in the *Loads* section.
- **rigid_body**: references a rigid body. Rigid bodies are defined implicitly via rigid materials.

All of these properties define arrays. A specific property can be referenced via square brackets. For instance, to reference the first material in the model, do this

```
fem.material[0]
```

Alternatively, a property can also be reference by name or by ID. In either case, use round brackets. For instance, to reference by ID, do

```
fem.material(1)
```

The ID refers to the “id” attribute that can be specified for most model components. To reference by name,

```
fem.material('MyMaterial')
```

Note the single quotes around the material name. The name is defined via the “name” attribute.

Note that rigid bodies can only be referenced via their material name.

```
fem.rigidbody('MyRigidMaterial')
```

Each property can have a list of parameters and sub-properties. To reference a parameter, just end the reference string with the name of the parameter. For example,

```
fem.material[0].E
```

Sub-properties are referenced similarly. For example, assume that the first material is a biphasic material that defines the solid property.

```
fem.material[0].solid.E
```

If the parameter itself is an array, a specific item in the array is referenced using square brackets. For example, for an EFD-neo Hookean material,

```
fem.material[0].ksi[0]
```

In cases where the model parameter does not define a scalar it is possible to reference the components of the parameter. For instance, a vec3 parameter defines an x, y, and z component.

```
fem.body_load[0].force.x
```

Nodal positions can also be referenced using the *mesh* property.

```
fem.mesh.node[0].position.x
```

In certain applications (e.g. optimization), it is also possible to reference output data, i.e. data that is normally used in the logfile section of the input file. This data is considered read-only. For example, the following references the 'sx' (xx-stress) of element with id 1.

```
fem.element_data('sx', 1)
```

Another example evaluates the enclosed volume of a surface (assuming the surface is closed).

```
fem.surface_data('volume', 'SomeSurface1')
```

Appendix C

Math Expression

Many model parameters can be defined via mathematical expression. This appendix documents the syntax of this capability.

C.1 Functions

The following functions are currently supported.

sin(x) The sine function.

cos(x) The cosine function

tan(x) The tangent function

csc(x) The cosecans function

sec(x) The secans function

cot(x) The cotangent function

acos(x) The arccosine function

asin(x) The arcsine function

cosh(x) The hyperbolic cosine function

sinh(x) The hyperbolic sine function

tanh(x) The hyperbolic tangent function

ln(x) The natural logarithm

log(x) The logarithm with base 10

exp(x) The exponential function

sqrt(x) The square root function

abs(x) The absolute value function

sgn(x) The sign function

H(x) The Heaviside function (x = 0.5 at zero)

step(x) The right-continuous unit step function

J0(x) Bessel function of the first kind, order 0

J1(x) Bessel function of the first kind, order 1

Jn(x,n) Bessel function of the first kind, order *n*

Y0(x) Bessel function of the second kind, order 0

Y1(x) Bessel function of the second kind, order 1

Yn(x,n) Bessel function of the second kind, order *n*

sinc(x) The *sinc* function (i.e. sin(x)/x)

fac(n) The factorial of *n*

erf(x) The *error* function

erfc(x) The complementary error function (erfc(x) = 1 - erf(x))

tgamma(x) The *Gamma* function

atan2(x,y) The two-parameter arctangent function

pow(x,y) The y-th power of x

Tn(n,x) The Chebyshev polynomial of order *n*

binomial(n,m) The binomial of (n,m)

max(x1,...,xn) Returns the maximum value of the values listed

min(x1,...,xn) Returns the minimum value of the values listed

avg(x1,...,xn) Returns the average value of the values listed

C.2 Constants

The following predefined constants are supported:

pi The value of pi (= 3.1415926535897932385)

e The value of exp(1) (= 2.7182818284590452354)

gamma The Euler-Mascheroni constant (= 0.57721566490153286061)

inf A very large number (= 1e308)

Bibliography

[1] G Aguilar, F Gaspar, F Lisbona, and C24558811158 Rodrigo. Numerical stabilization of biot's consolidation model by a perturbation on the flow equation. *International journal for numerical methods in engineering*, 75(11):1282–1300, 2008.

[2] Michael B Albro, Vikram Rajan, Roland Li, Clark T Hung, and Gerard A Ateshian. Characterization of the concentration-dependence of solute diffusivity and partitioning in a model dextran-agarose transport system. *Cell Mol Bioeng*, 2(3):295–305, Sep 2009.

[3] E.M. Arruda and M.C. Boyce. A three-dimensional constitutive model for the large stretch behavior of rubber elastic materials. *J. Mech. Phys. Solids*, 41(2):389–412, 1993.

[4] G. A. Ateshian. Anisotropy of fibrous tissues in relation to the distribution of tensed and buckled fibers. *J Biomech Eng*, 129(2):240–9, 2007.

[5] G. A. Ateshian and K. D. Costa. A frame-invariant formulation of fung elasticity. *J Biomech*, 42(6):781–5, 2009.

[6] G. A. Ateshian, V. Rajan, N. O. Chahine, C. E. Canal, and C. T. Hung. Modeling the matrix of articular cartilage using a continuous fiber angular distribution predicts many observed phenomena. *J Biomech Eng*, 131(6):061003, 2009.

[7] G. A. Ateshian and T. Ricken. Multigenerational interstitial growth of biological tissues. *Biomech Model Mechanobiol*, 9(6):689–702, 2010.

[8] G. A. Ateshian, W. H. Warden, J. J. Kim, R. P. Grelsamer, and V. C. Mow. Finite deformation biphasic material properties of bovine articular cartilage from confined compression experiments. *J Biomech*, 30(11-12):1157–64, 1997.

[9] G.A. Ateshian, M. B. Albro, S.A. Maas, and J.A. Weiss. Finite element implementation of mechanochemical phenomena in neutral deformable porous media under finite deformation. *Journal of Biomechanical Engineering*, 133(8):1005–1017, 2011.

[10] G.A Ateshian, Morrison B., J.W. Holmes, and C. T. Hung. Mechanics of cell growth. *Mechanics Research Communications*, 42:118–125, 2012.

[11] GA Ateshian, SA Maas, and J.A. Weiss. Finite element algorithm for frictionless contact of porous permeable media under finite deformation and sliding. *J. Biomech. Engn.*, 132(6):1006–1019, 2010.

[12] Gerard A Ateshian. Viscoelasticity using reactive constrained solid mixtures. *J Biomech*, 48(6):941–7, Apr 2015.

[13] Gerard A Ateshian, Benjamin J Ellis, and Jeffrey A Weiss. Equivalence between short-time biphasic and incompressible elastic material responses. *J Biomech Eng*, 129(3):405–12, Jun 2007.

[14] Gerard A Ateshian, Steve Maas, and Jeffrey A Weiss. Solute transport across a contact interface in deformable porous media. *J Biomech*, 45(6):1023–7, Apr 2012.

[15] Daniel Balzani, Sarah Brinkhues, and Gerhard A. Holzapfel. Constitutive framework for the modeling of damage in collagenous soft tissues with application to arterial walls. *Computer Methods in Applied Mechanics and Engineering*, 213-216:139–151, 2012.

[16] Klaus-Jürgen Bathe and Eduardo N Dvorkin. A formulation of general shell elements—the use of mixed interpolation of tensorial components. *International journal for numerical methods in engineering*, 22(3):697–722, 1986.

[17] P Betsch and E Stein. An assumed strain approach avoiding artificial thickness straining for a non-linear 4-node shell element. *Communications in Numerical Methods in Engineering*, 11(11):899–909, 1995.

[18] Anna M. Birzle, Christian Martin, Stefan Uhlig, and Wolfgang A. Wall. A coupled approach for identification of nonlinear and compressible material models for soft tissue based on different experimental setups - exemplified and detailed for lung parenchyma. *Journal of the Mechanical Behavior of Biomedical Materials*, 94:126–143, 2019.

[19] M Bischoff and E Ramm. Shear deformable shell elements for large strains and rotations. *International Journal for Numerical Methods in Engineering*, 40(23):4427–4449, 1997.

[20] Manfred Bischoff, E Ramm, and J Irslinger. Models and finite elements for thin-walled structures. *Encyclopedia of Computational Mechanics Second Edition*, pages 1–86, 2018.

[21] SS Blemker. *3D Modeling of Complex Muscle Architecture and Geometry*. Thesis, 2004.

[22] Javier Bonet and Richard D. Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge University Press, 1997.

[23] D R Carter and W C Hayes. Bone compressive strength: the influence of density and strain rate. *Science*, 194(4270):1174–6, Dec 1976.

[24] D R Carter and W C Hayes. The compressive behavior of bone as a two-phase porous structure. *J Bone Joint Surg Am*, 59(7):954–62, Oct 1977.

[25] Y I Cho and K R Kensey. Effects of the non-Newtonian viscosity of blood on flows in a diseased arterial vessel. Part 1: Steady flows. *Biorheology*, 28(3-4):241–62, 1991.

[26] JC Criscione, SA Douglas, and WC Hunter. Physically based strain invariant set for materials exhibiting transversely isotropic behavior. *J. Mech. Phys. Solids*, 49:871–897, 2001.

[27] John C Criscione, Jay D Humphrey, Andrew S Douglas, and William C Hunter. An invariant basis for natural strain which yields orthogonal stress response terms in isotropic hyperelasticity. *J. Mech. Phys. Solids*, 48(12):2445–2465, 2000.

[28] A. Curnier, He Qi-Chang, and P. Zysset. Conewise linear elastic materials. *J Elasticity*, 37(1):1–38, 1994.

[29] Daniel Charles Drucker. Relation of experiments to mathematical theories of plasticity. *Journal of Applied Mechanics*, 1949.

[30] Ana Cristina Estrada, Kyoko Yoshida, Samantha A Clarke, and Jeffrey W Holmes. Longitudinal reinforcement of acute myocardial infarcts improves function by transmurally redistributing stretch and stress. *J Biomech Eng*, 142(2), 02 2020.

[31] Y. C Fung. *Biomechanics: mechanical properties of living tissues*. Springer-Verlag, New York, 1981.

[32] Y. C. Fung. *Biomechanics : mechanical properties of living tissues*. Springer-Verlag, New York, 2nd edition, 1993.

[33] Y. C. Fung, K. Fronek, and P. Patitucci. Pseudoelasticity of arteries and the choice of its mathematical expression. *Am J Physiol*, 237(5):H620–31, 1979.

[34] Y. C Fung, Nicholas Perrone, and M Anliker. *Biomechanics, its foundations and objectives*. Prentice-Hall, Englewood Cliffs, N.J., 1972.

[35] T Christian Gasser, Ray W Ogden, and Gerhard A Holzapfel. Hyperelastic modelling of arterial layers with distributed collagen fibre orientations. *J R Soc Interface*, 3(6):15–35, Feb 2006.

[36] M.W. Gee, C.R. Dohrmann, S.W. Key, and W.A. Wall. A uniform nodal strain tetrahedron with isochoric stabilization. *Int. J. Numer. Meth. Engng*, (78):429–443, 2009.

[37] M.J.A. Girard, J.C. Downs, and C. F. Burgoyne. Peripapillary and posterior scleral mechanics - part i: Development of an anisotropic hyperelastic constitutive model. *J Biomech Eng*, 131(5):051011, 2009.

[38] C.L.M. Gouget, M.J.A. Girard, and C.R. Ethier. A constrained von mises distribution to describe fiber organization in thin soft tissues. *Biomechanics And Modeling in Mechanobiolgy*, 11(3-4):475–482, 2012.

[39] J.M. Guccione and A.D. McCulloch. Mechanics of active contraction in cardiac muscle: part i - constitutive relations for fiber stress that describe deactivation. *J. Biomechanical Engineering*, vol. 115(no. 1):72–83, 1993.

[40] Osman Gültekin, Hüsnü Dal, and Gerhard A Holzapfel. On the quasi-incompressible finite element analysis of anisotropic hyperelastic materials. *Computational mechanics*, 63(3):443–453, 2019.

[41] J Helfenstein, M Jabareen, Edoardo Mazza, and S Govindjee. On non-physical response in models for fiber-reinforced hyperelastic materials. *International Journal of Solids and Structures*, 47(16):2056–2061, 2010.

[42] M. H. Holmes and V. C. Mow. The nonlinear characteristics of soft gels and hydrated connective tissues in ultrafiltration. *J Biomech*, 23(11):1145–56, 1990.

[43] C Hou and G.A. Ateshian. A gauss-kronrod-trapezoidal integration scheme for modeling biological tissues with continuous fiber distributions. *Computer Methods in Biomechanics and Biomedical Engineering*, 19(8):883–893, 2016.

[44] Jay C Hou, Steve A Maas, Jeffrey A Weiss, and Gerard A Ateshian. Finite element formulation of multiphasic shell elements for cell mechanics analyses in febio. *Journal of Biomechanical Engineering*, 140(12), 2018.

[45] P J Hunter, A D McCulloch, and H E ter Keurs. Modelling the mechanical properties of cardiac muscle. *Prog Biophys Mol Biol*, 69(2-3):289–331, 1998.

[46] J. C. Iatridis, L. A. Setton, R. J. Foster, B. A. Rawlins, M. Weidenbaum, and V. C. Mow. Degeneration affects the anisotropic and nonlinear behaviors of human anulus fibrosus in compression. *J Biomech*, 31(6):535–44, 1998.

[47] David Kamensky, Fei Xu, Chung-Hao Lee, Jinhui Yan, Yuri Bazilevs, and Ming-Chen Hsu. A contact formulation based on a volumetric potential: Application to isogeometric simulations of atrioventricular valves. *Computer Methods in Applied Mechanics and Engineering*, 330:522–546, 2018.

[48] DE Kiousis, TC Gasser, and GA Holzapfel. Smooth contact strategies with emphasis on the modeling of balloon angioplasty with stenting. *International journal for numerical methods in engineering*, 75(7):826–855, 2008.

[49] Dimitrios E Kiousis, Alexander R Wulff, and Gerhard A Holzapfel. Experimental studies and numerical analysis of the inflation and interaction of vascular balloon catheter-stent systems. *Annals of Biomedical Engineering*, 37(2):315–330, 2009.

[50] S Klinkel, F Gruttmann, and W Wagner. A continuum based three-dimensional shell element for laminated structures. *Computers & Structures*, 71(1):43–62, 1999.

[51] W M Lai, J S Hou, and V C Mow. A triphasic theory for the swelling and deformation behaviors of articular cartilage. *J Biomech Eng*, 113(3):245–58, Aug 1991.

[52] Y. Lanir. Constitutive equations for fibrous connective tissues. *J Biomech*, 16(1):1–12, 1983.

[53] T. A. Laursen and B. N. Maker. Augmented lagrangian quasi-newton solver for constrained nonlinear finite element applications. *International Journal for Numerical Methods in Engineering*, 38(21):3571–3590, 1995.

[54] T. A. Laursen and J. C. Simo. Continuum-based finite element formulation for the implicit solution of multibody, large deformation frictional contact problems. *International Journal for Numerical Methods in Engineering*, 36(20):3451–3485, 1993.

[55] S.A. Maas, A. Erdemir, J.P. Halloran, and J.A. Weiss. A general framework for applications of prestrain to computational models of biological materials. *Journal of Biomechanical Behavior of Biomedical Materials*, 61:499–510, 2016.

[56] Richard H MacNeal. A simple quadrilateral shell element. *Computers & Structures*, 8(2):175–183, 1978.

[57] B. N. Maker. Rigid bodies for metal forming analysis with nike3d. *University of California, Lawrence Livermore Lab Rept*, UCRL-JC-119862:1–8, 1995.

[58] A Ya Malkin. Continuous relaxation spectrum-its advantages and methods of calculation. *Applied Mechanics and Engineering*, 11(2):235, 2006.

[59] Andreas Menzel and Ellen Kuhl. Frontiers in growth and remodeling. *Mech Res Commun*, 42:1–14, Jun 2012.

[60] V.C. Mow, S.C. Kuei, W.M. Lai, and C.G. Armstrong. Biphasic creep and stress relaxation of articular cartilage in compression: Theory and experiments. *J. Biomech. Eng.*, 102:73–84, 1980.

[61] Robert J Nims, Krista M Durney, Alexander D Cigan, Antoine Dusséaux, Clark T Hung, and Gerard A Ateshian. Continuum theory of fibrous tissue damage mechanics using bond kinetics: application to cartilage tissue engineering. *Interface Focus*, 6(1):20150063, Feb 2016.

[62] J T Overbeek. The donnan equilibrium. *Prog Biophys Biophys Chem*, 6:57–84, 1956.

[63] Ronald L Panton. *Incompressible flow*. John Wiley & Sons, 2006.

[64] Tasos C Papanastasiou. Flows of materials with yield. *Journal of Rheology*, 31(5):385–404, 1987.

[65] M. A. Puso and J. A. Weiss. Finite element implementation of anisotropic quasi-linear viscoelasticity using a discrete spectrum approximation. *J Biomech Eng*, 120(1):62–70, 1998.

[66] K. M. Quapp and J. A. Weiss. Material characterization of human medial collateral ligament. *J Biomech Eng*, 120(6):757–63, 1998.

[67] E K Rodriguez, A Hoger, and A D McCulloch. Stress-dependent finite growth in soft elastic tissues. *J Biomech*, 27(4):455–67, Apr 1994.

[68] Carlo Sansour. On the physical assumptions underlying the volumetric-isochoric split and the case of anisotropy. *European Journal of Mechanics-A/Solids*, 27(1):28–39, 2008.

[69] Lei Shi, Lingfeng Hu, Nicole Lee, Shuyang Fang, and Kristin Myers. Three-dimensional anisotropic hyperelastic constitutive model describing the mechanical response of human and mouse cervix. *Acta Biomaterialia*, 150:277–294, 2022.

[70] Jay J Shim, Steve A Maas, Jeffrey A Weiss, and Gerard A Ateshian. Finite element implementation of computational fluid dynamics with reactive neutral and charged solute transport in febio. *J Biomech Eng*, pages 1–26, May 2023.

[71] JC Simo. On a fully three-dimensional finite-strain viscoelastic damage model: formulation and computational aspects. *Computer methods in applied mechanics and engineering*, 60(2):153–173, 1987.

[72] JC Simo, F Armero, and RL Taylor. Improved versions of assumed enhanced strain tri-linear elements for 3d finite deformation problems. *Computer methods in applied mechanics and engineering*, 110(3-4):359–386, 1993.

[73] J.C. Simo and R.L. Taylor. Quasi-incompressible finite elasticity in principal stretches: Continuum basis and numerical algorithms. *Computer Methods in Applied Mechanics and Engineering*, 85:273–310, 1991.

[74] Juan C Simo and MS10587420724 Rifai. A class of mixed assumed strain methods and the method of incompatible modes. *International journal for numerical methods in engineering*, 29(8):1595–1638, 1990.

[75] RP Skelton, HJ Maier, and H-J Christ. The bauschinger effect, masing model and the ramberg–osgood relation for cyclic deformation in metals. *Materials Science and Engineering: A*, 238(2):377–390, 1997.

[76] Anthony James Merril Spencer. *Continuum Theory of the Mechanics of Fibre-Reinforced Composites*. Springer-Verlag, New York, 1984.

[77] A.M. Swedberg, S.P. Reese, S.A. Maas, B. J. Ellis, and J.A. Weiss. Continuum description of the poisson's ratio of ligament and tendon under finite deformation. *Journal of Biomechanics*, 47(12):3201–3209, 2014.

[78] D.R. Veronda and R.A. Westmann. Mechanical characterization of skin - finite deformations. *J. Biomechanics*, Vol. 3:111–124, 1970.

[79] Irene E Vignon-Clementel, C Alberto Figueroa, Kenneth E Jansen, and Charles A Taylor. Outflow boundary conditions for three-dimensional finite element modeling of blood flow and pressure in arteries. *Comput. Methods Appl. Mech. Engrg.*, 195(29):3776–3796, 2006.

[80] L Vu-Quoc and XG Tan. Optimal solid shells for non-linear analyses of multilayer composites. i. statics. *Computer methods in applied mechanics and engineering*, 192(9-10):975–1016, 2003.

[81] H Weinans, R Huiskes, and H J Grootenboer. The behavior of adaptive bone-remodeling simulation models. *J Biomech*, 25(12):1425–41, Dec 1992.

[82] J.A. Weiss, J.C. Gardiner, and Bonifasi-Lista C. Ligament material behavior is nonlinear, viscoelastic and rate-independent under shear loading. *Journal of Biomechanics*, 35(7):943–950, 2002.

[83] J.A. Weiss, B.N. Maker, and S. Govindjee. Finite element implementation of incompressible, transversely isotropic hyperelasticity. *Computer Methods in Applications of Mechanics and Engineering*, 135:107–128, 1996.

[84] Brandon K. Zimmerman, David Jiang, Jeffrey A. Weiss, Lucas H. Timmins, and Gerard A. Ateshian. On the use of constrained reactive mixtures of solids to model finite deformation isothermal elastoplasticity and elastoplastic damage mechanics. *Journal of the Mechanics and Physics of Solids*, page 104534, 2021.