

FEBio

FINITE ELEMENTS FOR BIOMECHANICS

Version 1.5

User's Manual

Last Updated: May 23, 2012

Contributors

Steve Maas (steve.maas@utah.edu)
Dave Rawlins (rawlins@sci.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://mrl.sci.utah.edu/software/febio>

Forum

<http://mrl.sci.utah.edu/forums/>

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



Table of Contents

CHAPTER 1 INTRODUCTION.....	1
1.1. OVERVIEW OF FEBIO.....	1
1.2. ABOUT THIS DOCUMENT	2
CHAPTER 2 RUNNING FEBIO	3
2.1. RUNNING FEBIO ON WINDOWS	3
2.1.1. Windows XP	3
2.1.2. Windows 7.....	3
2.2. RUNNING FEBIO ON LINUX OR MAC	4
2.3. THE COMMAND LINE	4
2.4. THE FEBIO PROMPT	7
2.5. THE CONFIGURATION FILE	7
2.6. USING MULTIPLE PROCESSORS	7
2.7. FEBIO OUTPUT	8
2.8. ADVANCED OPTIONS	8
2.8.1. Interrupting a run.....	8
2.8.2. Debugging a run	9
2.8.3. Restarting a run	10
2.8.4. Input file checking.....	10
CHAPTER 3 FREE FORMAT INPUT	11
3.1. FREE FORMAT OVERVIEW	12
3.2. MODULE SECTION.....	13
3.3. CONTROL SECTION	14
3.3.1. Common Parameters.....	14
3.3.2. Parameters for poro analysis.....	20
3.3.3. Parameters for solute analysis.....	20
3.3.4. Parameters for heat analysis.....	20
3.4. MATERIAL SECTION.....	21
3.5. GEOMETRY SECTION.....	22
3.5.1. Nodes Section.....	22
3.5.2. Elements Section	22
3.5.2.1. Solid Elements	22
3.5.2.2. Shell Elements.....	23
3.5.2.3. Surface Elements.....	24
3.5.3. ElementData Section.....	24
3.6. INITIAL SECTION	26
3.6.1. Initial Nodal Velocities	26
3.6.2. Initial Nodal Effective Fluid Pressure.....	26
3.6.3. Initial Nodal Effective Concentration	26
3.7. BOUNDARY SECTION	27
3.7.1. Prescribed Nodal Degrees of Freedom.....	27
3.7.2. Fixed Nodal Degrees of Freedom.....	28
3.7.3. Contact Interfaces.....	28
3.7.3.1. Sliding Interfaces	28
3.7.3.2. Biphasic Contact	33
3.7.3.3. Biphasic-Solute Contact.....	34
3.7.3.4. Rigid Wall Interfaces	34
3.7.3.5. Tied Interfaces.....	35
3.7.3.6. Rigid Interfaces.....	36
3.7.3.7. Rigid Joints	36

3.7.4. Spring section.....	36
3.8. LOADS SECTION.....	38
3.8.1. Nodal Loads.....	38
3.8.2. Pressure forces.....	38
3.8.3. Mixture Normal Traction.....	39
3.8.4. Fluid Flux.....	40
3.8.5. Solute Flux.....	41
3.8.6. Body Force.....	42
3.9. CONSTRAINTS SECTION.....	44
3.9.1. Rigid Body Constraints.....	44
3.10. GLOBALS SECTION.....	46
3.10.1. Constants.....	46
3.10.2. Solutes.....	46
3.11. LOADDATA SECTION.....	47
3.12. OUTPUT SECTION.....	49
3.12.1. Logfile.....	49
3.12.1.1. Node_Data class.....	51
3.12.1.2. Element_Data Class.....	52
3.12.1.3. Rigid_Body_Data Class.....	53
3.12.2. Plotfile.....	54
CHAPTER 4 MATERIALS.....	56
4.1. ELASTIC SOLIDS.....	56
4.1.1. Specifying fiber orientation.....	56
4.1.1.1. Transversely Isotropic materials.....	56
4.1.1.2. Orthotropic materials.....	58
4.1.2. Uncoupled Materials.....	60
4.1.2.1. Arruda-Boyce.....	62
4.1.2.2. Ellipsoidal Fiber Distribution.....	63
4.1.2.3. Ellipsoidal Fiber Distribution Mooney-Rivlin.....	65
4.1.2.4. Ellipsoidal Fiber Distribution Veronda-Westmann.....	66
4.1.2.5. Fiber with Exponential-Power Law, Uncoupled Formulation.....	67
4.1.2.6. Fung Orthotropic.....	69
4.1.2.7. Mooney-Rivlin.....	70
4.1.2.8. Muscle Material.....	71
4.1.2.9. Ogden.....	73
4.1.2.10. Tendon Material.....	74
4.1.2.11. Tension-Compression Nonlinear Orthotropic.....	75
4.1.2.12. Transversely Isotropic Mooney-Rivlin.....	76
4.1.2.13. Transversely Isotropic Veronda-Westmann.....	78
4.1.2.14. Uncoupled Solid Mixture.....	79
4.1.2.15. Veronda-Westmann.....	80
4.1.3. Compressible Materials.....	81
4.1.3.1. Donnan Equilibrium Swelling.....	83
4.1.3.2. Ellipsoidal Fiber Distribution.....	85
4.1.3.3. Ellipsoidal Fiber Distribution Neo-Hookean.....	87
4.1.3.4. Ellipsoidal Fiber Distribution with Donnan Equilibrium Swelling.....	88
4.1.3.5. Fiber with Exponential-Power Law.....	89
4.1.3.6. Holmes-Mow.....	91
4.1.3.7. Isotropic Elastic.....	92
4.1.3.8. Orthotropic Elastic.....	93
4.1.3.9. Neo-Hookean.....	94
4.1.3.10. Ogden Unconstrained.....	95
4.1.3.11. Perfect Osmometer Equilibrium Osmotic Pressure.....	96
4.1.3.12. Solid Mixture.....	98
4.1.3.13. Spherical Fiber Distribution.....	99
4.2. VISCOELASTIC SOLIDS.....	102
4.2.1. Uncoupled Viscoelastic Materials.....	102
4.2.2. Compressible Viscoelastic Materials.....	103

4.3. BIPHASIC MATERIALS.....	104
4.3.1. <i>General Specification of Biphasic Materials</i>	105
4.3.2. <i>Permeability Materials</i>	106
4.3.2.1. Constant Isotropic Permeability	107
4.3.2.2. Holmes-Mow	108
4.3.2.3. Referentially Isotropic Permeability.....	109
4.3.2.4. Referentially Orthotropic Permeability	110
4.3.2.5. Referentially Transversely Isotropic Permeability	112
4.4. BIPHASIC-SOLUTE MATERIALS	114
4.4.1. <i>Guidelines for Biphasic-Solute Analyses</i>	116
4.4.1.1. Prescribed Boundary Conditions.....	116
4.4.1.2. Prescribed Initial Conditions	116
4.4.2. <i>General Specification of Biphasic-Solute Materials</i>	117
4.4.3. <i>Diffusivity Materials</i>	119
4.4.3.1. Constant Isotropic Diffusivity	119
4.4.3.2. Constant Orthotropic Diffusivity.....	120
4.4.3.3. Referentially Isotropic Diffusivity	121
4.4.3.4. Referentially Orthotropic Diffusivity	122
4.4.4. <i>Solubility Materials</i>	124
4.4.4.1. Constant Solubility.....	124
4.4.5. <i>Osmotic Coefficient Materials</i>	125
4.4.5.1. Constant Osmotic Coefficient	125
4.5. TRIPHASIC MATERIALS	126
4.5.1. <i>Guidelines for Triphasic Analyses</i>	129
4.5.1.1. Initial State of Swelling.....	129
4.5.1.2. Prescribed Boundary Conditions.....	130
4.5.1.3. Prescribed Initial Conditions	130
4.5.1.4. Prescribed Electric Current Density	130
4.5.2. <i>General Specification of Triphasic Materials</i>	132
4.6. RIGID BODY	135
CHAPTER 5 RESTART INPUT FILE	136
5.1. THE ARCHIVE SECTION	136
5.2. THE CONTROL SECTION	137
5.3. THE LOADDATA SECTION	137
5.4. EXAMPLE	137
CHAPTER 6 MULTI-STEP ANALYSIS	138
6.1. THE STEP SECTION	138
6.1.1. <i>Control Settings</i>	139
6.1.2. <i>Boundary and Loads Settings</i>	139
6.2. AN EXAMPLE	139
CHAPTER 7 PARAMETER OPTIMIZATION	142
7.1. OPTIMIZATION INPUT FILE	142
7.1.1. <i>Model section</i>	142
7.1.2. <i>Options section</i>	142
7.1.3. <i>Function section</i>	143
7.1.4. <i>Parameters Section</i>	144
7.1.5. <i>LoadData Section</i>	145
7.2. RUNNING A PARAMETER OPTIMIZATION.....	145
7.3. AN EXAMPLE INPUT FILE	146
APPENDIX A. CONFIGURATION FILE.....	147
APPENDIX B. LSDYNA DATABASE FORMAT	148
REFERENCES	150

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. This chapter briefly describes the available features of FEBio. All features can be used together seamlessly, giving the user a powerful tool for solving 3D problems in computational biomechanics. The software is open-source, and pre-compiled executables for Windows, OS-X and Linux platforms are available for download at <http://mrl.sci.utah.edu/software>.

FEBio supports two analysis types, namely *quasi-static* and *dynamic*. In a *quasi-static* analysis, the (quasi-) static response of the system is sought and the effects of inertia are ignored. In the presence of biphasic materials, a coupled solid-fluid problem is solved. In a dynamic analysis, the inertial effects are included in the governing equations to calculate the time dependent response of the system.

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 constitutive models available. These models exhibit anisotropic behavior in a single preferred direction and are useful for representing biological tissues such as tendons, muscles 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 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. The formulation depends on the element type. For hexahedral elements a three-field formulation is used, and for tetrahedral elements a nodally integrated formulation is implemented. These algorithms allow the user to capture the accurate response of highly incompressible materials.

FEBio supports a wide range of boundary conditions to model interactions between materials that are relevant to problems in biomechanics. These include prescribed displacements, nodal forces and pressure forces. 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 and rigid walls, are available as well. As of version 1.2 it is also possible to model the fluid flow across two contacting poroelastic materials. Finally, the

user may specify a body force to model the effects such as, gravity, base acceleration or centripetal acceleration.

FEBio is a nonlinear implicit FE solver and does not have mesh generation capabilities. Therefore the input files, which are described in detail in this document, need to be generated by preprocessing software. The preferred preprocessor for FEBio is called [PreView](#). PreView can convert some other formats to the FEBio input specification. For instance, NIKE3D [1] and Abaqus input files can be imported to PreView and can be exported from PreView as a FEBio input file. See the PreView [User's Manual](#) for more information

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 [Theory Manual](#), which describes the theory behind the FEBio algorithms.

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.

Although this document describes some of the theoretical aspects of FEBio, a complete theoretical development can be found in the [FEBio Theory Manual](#). Developers who are interested in modifying or extending the FEBio code will find the [FEBio Developer's Manual](#) very useful.

Chapter 2 Running FEBio

FEBio runs on several different computing platforms including Windows XP, Mac OSX and most 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. The process is slightly different depending on whether you are using Windows XP or Windows 7, so we'll look at the two Windows versions separately.

2.1.1. Windows XP

First, we'll add the FEBio installation folder to the PATH variable. Open the *Control Panel* from the *Start* menu. Switch to *Classic View* and double-click the *System* icon. In the dialog box that appears, select the *Advanced* tab and click the button named *Environment variables*. Find the *path* variable and click the *Edit* button. Add the end of the PATH's value (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/). Then click the *OK*-button on all open dialog boxes.

To open a command prompt, click the *Run* menu item on the *Start* menu. In the dialog box that appears type *cmd* and press the *OK*-button. A command prompt window appears. You can now use the *cd* (change directory) command to navigate to the folder that contains the FEBio input files. To run FEBio, simply type *febio* (with or without additional arguments) and press *Enter*.

2.1.2. Windows 7

Let's first modify the *PATH* environment variable. Open the *Start* menu and type *system* in the search field. From the search results, select the *System* option under *Control Panel*. The *System* window will appear. Find the *Change Settings* option (on the lower, right side) and click it. The *System Properties* dialog box appears. Activate the *Advanced* tab and click the *Environment Variables* button. Find the *path* variable and click the *Edit* button. Add the end of the PATH's value (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/). Then click the *OK*-button on all open dialog boxes.

Next, open a command prompt as follows. Click the *Start* menu and type *cmd* in the search field. From the search results, select the *cmd* option under Programs. A command prompt window

appears. You can now use the *cd* (change directory) command to navigate to the folder that contains the FEBio input files. To run FEBio, simply type *febio* (with or without additional arguments) and press *Enter*.

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 [name1] | -o2 [name2] | ...]
```

Where *-o1*, *-o2* are options and *name1*, *name2*, ... are filenames. The different options (of which most are optional) are given by the following list:

- *-i* : name of input file
- *-r* : restart file name
- *-g* : debug flag (does not require a file name)
- *-p*: plot file name
- *-a*: dump file name
- *-o*: log file name
- *-c*: data check only
- *-s*: material parameter optimization
- *-nosplash*: don't show the welcome screen
- *-cnf*: configuration filename
- *-noconfig*: don't use the configuration file

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
```

-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 Chapter 5. The `-i` and `-r` options are mutually exclusive; only one of them may appear on the command line.

Example:

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

-g

The `-g` option runs FEBio in *debug mode*. 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.plt
```

-a

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 `-a` command line option allows the user to override the default name for the dump file. See section 2.8.3 and Chapter 5 for more details on how to use the restart feature.

Example:

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

-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
```

-c

When the `-c` option is specified on the command line, FEBio will only read the input file and check it for possible errors. When the check is complete, FEBio will terminate. See section 2.8.4 for more details on this option.

Example:

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

-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
```

-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 -nosplash
```

-cnf

-noconfig

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 `-cnf` 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 Appendix A.

Example:

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

-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 mutually exclusive with `-i` and `-r`. Also, the restart feature does not work with the optimization module.

Example:

```
> febio -s file.feb
```

2.4. The FEBio prompt

If you start FEBio without any command arguments, the FEBio prompt will appear (after the welcome message). It will look something like this:

```
febio>
```

You can now enter one of the following commands:

- **help**: prints an overview of available commands with a brief description of each command.
- **quit**: exit FEBio.
- **run**: run an FEBio input file. This command takes the same options as 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
```

- **version**: print version information.

You can also bring up the FEBio prompt during a run by pressing ctrl+c. See section 2.8.1 for more details.

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. The configuration file uses an xml format to store data and is detailed in Appendix A. 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 configuration file needs to be stored in the same location as the executable. Alternatively, the location of the file can also be specified on the command line using the `-cnf` option.

Example:

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

2.6. Using multiple processors

Versions 1.2 and later of FEBio implement the [MKL](#) version of the [PARDISO](#) linear solver, which can run on multiple processors. Set the environment variable `OMP_NUM_THREADS` to the number of threads to be used. For example, on Linux using the Bash shell, execute:

Example:

```
> export OMP_NUM_THREADS=4
```

Using the c shell execute:

Example:

```
> setenv OMP_NUM_THREADS 4
```

Or at a Windows command prompt:

Example:

```
> set OMP_NUM_THREADS=4
```

2.7. FEBio output

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 [PostView](#). 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 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.plt* and the dump file is called *input.dmp*.

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 following commands are available.

- *cont*: continues the run. FEBio will continue the analysis where it left off.
- *conv*: force the current time step to converge. This is useful for example when a time step is having difficulty satisfying too tight of convergence criteria. The user can then manually force the convergence of the time 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.

^{*} This feature may not work properly on all systems, although it will always work on Windows systems.

- *debug [on/off]*: entering *debug* will toggle debug mode. 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.
- *dtmin*: set the minimum time step size. This command overrides the minimum time step size that was specified in the input file.
- *fail*: stop the current iteration and retry. 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.
- *help*: list the available commands with a short explanation. Prints the information provided in this section of the manual.
- *plot*: dump current state to plot database and continue. This 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.
- *print*: print values of variables:
 - *nnz*: number of non-zeroes in stiffness matrix
 - *neq*: number of equations
 - *time*: the current time step
- *quit*: exit the application
- *restart*: toggles restart flag. 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 Chapter 5 for more details on FEBio's restart feature.
- *time*: print elapsed time and an estimation of the remaining time.
- *version*: print version information

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 may be in the middle of a call to the linear solver or another time-consuming part of the analysis procedure.

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 geometry 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. Restarting a run

When the creation of a restart 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 a restart file, simply set the appropriate option in the control section of the input file. This will generate a *dump* file which then can be used to restart the analysis. See Chapter 5 for more details.

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.

2.8.4. Input file checking

The `-c` option allows the user to stop FEBio after the initial data checking is done. This way, potential input errors can be spotted without running the actual problem.

Example:

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

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.

```
<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.

```
<!-- 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. This document describes version 1.2 of the FEBio specification[†]. The root element will therefore be defined as follows:

```
<febio_spec version="1.2">
  <!-- 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:

- *Control*: specifies control and solver parameters.
- *Material*: Specifies the materials used in the problem and the material parameters.
- *Geometry*: Defines the geometry of the problem, such as nodal coordinates and element connectivity.
- *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.
- *Globals*: Defines the global variables in the scene, e.g. body force.
- *LoadData*: Defines the load curve data.
- *Ouput*: Defines additional data that is to be stored.
- *Step*: Defines different analysis steps, where in each analysis the boundary and initial conditions can be redefined.

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

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
poro	Biphasic analysis: steady-state or transient
solute	Multiphasic analysis including solute transport: steady-state or transient
heat	Heat transfer analysis: steady-state or transient

If the Module section is omitted, it is assumed that the analysis is a *solid* analysis, i.e. a quasi-static or dynamic structural mechanics problem.

Important Note: Previous versions of FEBio 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="poro"/>
  <!-- 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. The parameters depend somewhat on the analysis as defined by the Module section. Many parameters are common to all types of analysis, so they are listed first.

3.3.1. Common Parameters

The following parameters are common for all analysis. 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
title	Title of problem	(none)
time_steps*	Total number of time steps. (= <i>ntime</i>)(1)	(none)
step_size*	The initial time step size. (= <i>dt</i>) (1)	(none)
dtol	Convergence tolerance on displacements (2)	0.001
etol	Convergence tolerance on energy (2)	0.01
rtol	Convergence tolerance on residual (2)	0 (disabled)
lstol	Convergence tolerance on line search (3)	0.9
time_stepper	Enable the auto time stepper (4)	(off)
max_refs	Max number of stiffness reformations (5)	15
max_ups	Max number of BFGS stiffness updates (5)	10
optimize_bw	Optimize bandwidth of stiffness matrix (6)	0
restart	Generate restart flag (7)	0
plot_level	Sets the level of state dumps to the plot file (8)	PLOT_MAJOR_ITRS
cmax	Set the max condition number for the stiffness matrix (9)	1e+5
analysis	Sets the analysis type (10)	static
print_level	Sets the amount of output that is generated on screen (11)	PRINT_MINOR_ITRS
linear_solver	Set the linear solver (12)	skyline
min_residual	Sets minimal value for residual tolerance (13)	1e-20
integration	Set the integration rule for a particular element (14)	N/A

Comments:

1. The total running 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 running time will always be determined by $ntime * dt$.

2. 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.

3. 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.
4. If the *time_stepper* parameter is defined it will enable the auto time-stepper, which will adjust the new time step size based on convergence information from the previous time step. The following sub-elements may also be defined, although all are optional. Note that these are sub-elements of the *time_stepper* element and not of the *Control* element.

Parameter	Description	Default
dtmin	Minimum time step size	dt/3
dtmax	Maximum time step size	dt*3
max_retries	Maximum nr. of retries allowed per time step	5
opt_iter	Optimal number of iterations	10

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.

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 the *max_retries* parameter. For example, if the last time step size is 0.1 and *max_retries* is set to 5, 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.

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>
...
<loadcurve id="1" type="step">
  <loadpoint>0,0</loadpoint>
  <loadpoint>0.5, 0.1</loadpoint>
  <loadpoint>1.0, 0.2</loadpoint>
</loadcurve>
```

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.

5. The *max_ups* and *max_refs* parameters control the BFGS method that FEBio uses to solve the nonlinear FE equations. In this method the global stiffness matrix is only calculated 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, 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 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.

Note that when `max_ups` is set to 0, FEBio will use the Full-Newton method instead of the BFGS method. In other words, the stiffness matrix is reformed for every 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.

- 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.

- The `restart` parameter can be used to generate a restart dump file. To activate it, specify a non-zero value. A filename may be specified as an option. If the filename is omitted, a default name will be provided. Note that this will only generate the binary dump file that is needed to restart the analysis. To override certain parameters before restarting, create a restart input file. FEBio does not generate this file automatically so the user needs to create that file manually. See chapter 4 on the format of the restart input file.

```
<restart file="out.dmp">1</restart>
```

- 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.

- 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>
```

10. The *analysis* element sets the analysis type. Currently, FEBio defines three analysis types: (quasi-)static, steady-state, and dynamic. 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, or triphasic analysis
dynamic	dynamic analysis.

11. 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 equil. iterations

12. The linear solver is by default set in the configuration file. Since this allows FEBio to run the same file on different platforms (which may support different linear solvers), this is the recommended way to set the linear solver. However, if the need arises, the user can also override the default linear solver, by explicitly specifying the *linear_solver* parameter in the FEBio input file. The following linear solvers are implemented although they might not all be supported on all platforms. The only solver that is guaranteed to work on all platforms is the skyline solver.

Name	Description	Sym	ASym	MT
<i>skyline</i>	The default skyline solver	●		
<i>pardiso</i>	The preferred solver for most platforms	●	●	●
<i>superlu</i>	The single-threaded SuperLU solver	●	●	
<i>superlu_mt</i>	The multi-threader SuperLU solver	●	●	●
<i>wsmp</i>	The WSMP solver	●	●	●
<i>lusolver</i>	Full-matrix solver	●		

13. 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.

14. You can override FEBio’s default integration rule for specific element classes. For each element class, define a *rule* element in which you set the integration rule.

```
<integration>
  <rule elem="<elem>">VALUE</rule>
  <!-- repeat rules for other elements -->
</integration>
```

The *elem* attribute value defines for which element class you wish to override the default integration rule and can have any of the following values.

elem	Description
hex8	8-node hexahedral element
tet4	4-node tetrahedral element

The values of the rule elements depend on the *elem* attribute. for the *hex8* element, the following values are defined. The values marked with an asterisk (*) are the default.

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

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.
UT4	Nodally integrated tetrahedron.

The UT4 is a special formulation for tetrahedral elements that uses a nodally averaged integration rule, as proposed by Gee et al [2]. This formulation requires additional parameters. To override the default values, use the following alternative syntax:

```
<rule elem="tet4" type="UT4">
  <alpha>0.05</alpha>
  <iso_stab>0</iso_stab>
</rule>
```

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 Theory Manual for a detailed description of this formulation.

3.3.2. Parameters for poro analysis

A biphasic analysis is defined by using the *poro* 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
<code>ptol</code>	Specify the fluid pressure convergence tolerance	0.01
<code>symmetric_biphasic</code>	Choose between a symmetric or nonsymmetric formulation	1

3.3.3. Parameters for solute analysis

When the type attribute of the Module section is set to *solute* or *triphasic*, 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
<code>ctol</code>	Specify the concentration convergence tolerance	0.01

3.3.4. 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.

3.4. 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.

3.5. Geometry Section

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

- *Nodes*: contains nodal coordinates.
- *Elements*: contains element connectivity.
- *ElementData*: contains additional element data.

3.5.1. Nodes Section

The nodes section contains all the nodal coordinates. Repeat the following XML-element for each node:

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

The *id* attribute is a unique number that identifies the node. This *id* is used as a reference in the element connectivity section. The *ids* do not have to be in order, but the lowest *id* *must* be 1 and numbers cannot be skipped. (So if there is a node 4 and a node 6, there must also be a node 5 somewhere).

3.5.2. Elements Section

The element section contains all the element connectivity data. FEBio classifies elements in two classes, namely *solids* and *shells*.

3.5.2.1. Solid Elements

The following solid element types are defined:

hex8: 8-node hexahedral element
penta6: 6-node pentahedral element
tet4: 4-node tetrahedral element

The value for the XML element is the nodal connectivity:

```
<hex8 id="n" mat="m">n1,n2,n3,n4,n5,n6,n7,n8</hex8>
<penta6 id="n" mat="m">n1,n2,n3,n4,n5,n6</penta6>
<tet4 id="n" mat="m">n1,n2,n3,n4</tet4>
```

Elements have two attributes. First, there is a unique *id* that can be used to reference the element. The same limitations to this *id* apply as to the nodal *ids*. The second attribute is the material number. This number is the material *id* that was defined in the material section.

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

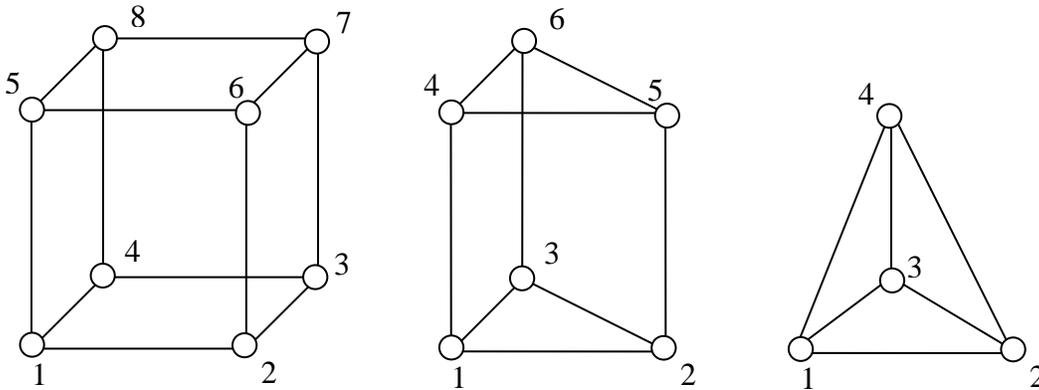


Figure 3-1. Node numbering for solid elements.

3.5.2.2. Shell Elements

FEBio currently supports a 4-noded quadrilateral and a 3-noded triangular shell element. Use the following XML tags to define the shell element:

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

Like the solid element, the shell element needs two attributes. The first one identifies the unique element ID, while the second is the material number. The shell thickness is specified in the ElementData section.

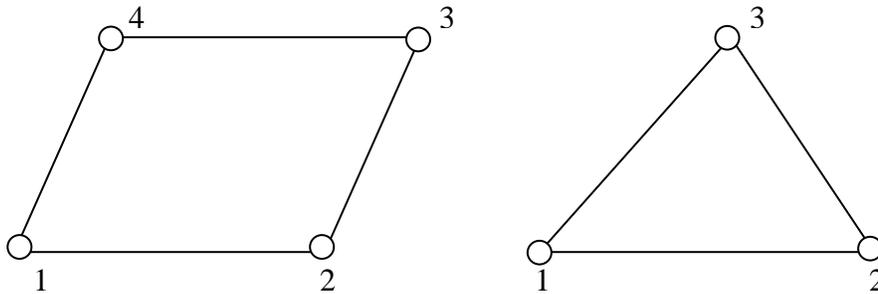


Figure 3-2. Node numbering for shell elements.

3.5.2.3. Surface Elements

In many cases the surface of some geometry (or part of it) is required. For example, pressure forces are applied to the surface. For this reason surface elements have to be defined. Two surface elements are available:

- *quad4*: 4-node quadrilateral element
- *tri3*: 3-node 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.

3.5.3. ElementData Section

Additional element data can be specified in the *ElementData* section. The data in this section usually represents material data that can vary from element to element. The following element properties can be defined:

Property	Description
fiber	Specify a local fiber direction
thickness	Specify the shell element thickness

Note that defining a *fiber* direction here will override the fiber distribution specified in the material definition and only for those elements specified in the *ElementData* section. In other words, you can define a particular fiber distribution in the material section and then override the fiber direction for a subset of elements. See section 4.1.1 for more information on defining fiber directions for transversely isotropic materials.

The thickness parameter can only be specified for shell elements. The value of this parameter is the shell thickness at each of the shell nodes. For example, for a four-node shell:

```
<element id="n">
  <thickness>0.01,0.01,0.01,0.01</thickness>
</element>
```

To specify a fiber direction for a particular element, enter the following xml-element in the *ElementData* section:

```
<element id="n">
  <fiber>1,0,0</fiber>
</element>
```

The *id* attribute identifies the element to which this fiber direction is applied. The *fiber* element defines a vector in global coordinates. This vector does not have to be of unit length since it is automatically normalized by FEBio.

3.6. Initial Section

The *Initial* section defines all initial conditions that may be applied to the analysis. Several initial conditions are available: nodal velocities, nodal effective pressure, and nodal effective concentrations.

3.6.1. Initial Nodal Velocities

Initial nodal velocities are needed in dynamic analyses (see Section 3.3.1). They can be prescribed using the *velocity* sub-element.

Example:

```
<Initial>
  <velocity>
    <node id="1"> 0.5, 1.0, 0</node>
    ...
  </velocity>
</Initial>
```

3.6.2. Initial Nodal Effective Fluid Pressure

Initial nodal effective fluid pressures are needed in biphasic-solute and triphasic analyses. They can be prescribed using the *fluid_pressure* sub-element.

Example:

```
<Initial>
  <fluid_pressure>
    <node id="1"> -2.477e-3</node>
    ...
  </fluid_pressure>
</Initial>
```

3.6.3. Initial Nodal Effective Concentration

Initial nodal effective solute concentrations are needed in biphasic-solute and triphasic analyses. They can be prescribed using the *concentration* sub-element. An optional *sol* attribute may be provided to identify the solute to which this initial condition applies, referencing the corresponding list in the Globals section (Section 3.10.2). If not specified, the default is 1.

Example:

```
<Initial>
  <concentration sol="1">
    <node id="1"> 1.0</node>
    ...
  </concentration>
</Initial>
```

3.7. Boundary Section

The *Boundary* section defines all fixed, prescribed and contact boundary conditions that may be applied to the geometry.

3.7.1. Prescribed Nodal Degrees of Freedom

Nodal degrees of freedom (dof) can be prescribed using the *prescribe* sub-element.

```
<prescribe [type="relative"]>
  <node id="n" bc="x" [lc="1"]>2.3</node>
  ...
</prescribe>
```

The optional *type* attribute in the *prescribe* tag allows users to choose between absolute (default) and relative boundary conditions. Absolute boundary conditions assign the specified value to the desired nodal degree of freedom. Relative boundary conditions are meaningful only in multi-step analyses. When a nodal degree of freedom is specified to be relative at a particular step, the value prescribed for that node is superposed over the value of that degree of freedom at the end of the preceding step.

The *id* attribute in the *node* tag indicates to which node this prescribed dof is applied.

The *bc* attribute in the *node* tag specifies the particular degree of freedom. The following values are allowed:

- x: apply displacement in *x*-direction
- y: apply displacement in *y*-direction
- z: apply displacement in *z*-direction
- p: apply prescribed effective fluid pressure
- cn: apply prescribed effective solute concentration on solute *n*
- t: apply prescribed temperature (heat transfer analysis)

For solutes, replace “*n*” with the solute id from the global solute table (Section 3.10.2); for example, “*c2*”. An optional loadcurve can be specified with the *lc* attribute in the *node* tag. The value of the *lc* attribute is the ID of the loadcurve that is defined in the *LoadData* section of the input file. If a loadcurve is not specified, the value will be automatically ramped from a value of 0 at time $t=0$ to the value specified in the xml file at the time corresponding to the end of the analysis (e.g. 2.3 in the example above).

The value of the *node* element (e.g. 2.3 in the example above) is the value for the prescribed displacement. Note that if a loadcurve is specified, this value scales the value determined by the loadcurve.

3.7.2. Fixed Nodal Degrees of Freedom

Degrees of freedom that are fixed (in other words, constrained, or are always zero) can be defined using the *fix* element:

```
<fix>
  <node id="n" bc="x"/>
  ...
</fix>
```

Although the *prescribe* element with a value of zero for the *node* tags can also be used to fix a certain nodal degree of freedom, the user should use the *fix* element 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 reduces the run time of the FE analysis.

3.7.3. Contact Interfaces

Contact boundary conditions are defined with the *contact* element. The *type* attribute specifies the type of contact interface that is defined. For example:

```
<contact type="sliding_with_gaps"> ... </contact>
```

The *type* can be one of the following options:

Type	Description
sliding_with_gaps, facet-to-facet sliding, sliding2, sliding3	A sliding interface that may separate (with biphasic contact for <i>sliding2</i> , and biphasic-solute contact for <i>sliding3</i>)
rigid_wall	A sliding interface with rigid wall as master surface
rigid	A rigid interface
rigid_joint	A joint between two rigid bodies
tied	A tied interface

3.7.3.1. 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_with_gaps (SWG)***: This is FEBio's original implementation of sliding contact. It is based on Laursen's contact formulation [3] 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.
- ***facet-to-facet sliding (F2F)***: This implementation is identical to the *sliding_with_gaps* 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.

- ***sliding2* (S2)**: This method is similar to the *facet-to-facet sliding* but differs in the linearization of the contact forces, which results in a different contact stiffness matrix compared to the previous two methods. This method is described in detail in [4]. 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. This particular contact implementation also supports biphasic contact (see the next section).
- ***sliding3* (S3)**: This method is similar to *sliding2*. This contact implementation supports biphasic-solute contact (see the next section). When using biphasic-solute materials, the non-symmetric version must be used.

The following table lists the properties that are defined for sliding interfaces. 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	SWG	F2F	S2/S3	Default
penalty	normal penalty scale factor (1)	●	●	●	1.0
auto_penalty	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)
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)
symmetric_stiffness	symmetric stiffness matrix flag (7)			●	0
search_tol	Projection search tolerance (8)	●	●	●	

The slave and master surfaces, which define the contact interface, are entered by specifying the *surface* element. The *type* attribute is used to specify whether it is a *slave* or *master* surface. The *surface* tag is followed by the definition of the surface elements. For example, a list of facets composing the master surface of the sliding interface could be written as:

```
<surface type="master">
  <quad4 id="n">n1,n2,n3,n4</quad4>
```

...
</surface>

Both quadrilateral surface elements (quad4) and triangular elements (tri3) can be used to define the master and/or slave surfaces.

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 ε_i for a particular node/integration point 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.

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 master surface and a slave surface. The slave surface is the surface over which the contact equations are integrated and on which the contact tractions are calculated. The master 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 slave and master surfaces. It is best to use the most tessellated surface as the slave and the coarsest as the master 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 slave and

master surfaces are swapped and another pass is performed. When using the two-pass method, the definition of master and slave surfaces is arbitrary. In most problems, the single pass is sufficient to enforce contact; with a judicious choice of slave-master 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 master 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 slave surface, the reaction forces may not propagate correctly to the master 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>
```

[‡] In future versions of FEBio rigid surfaces will be automatically picked to be the master.

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, $(\sqrt{\sum_i \langle g_i \rangle^2} / N)$, $\langle \bullet \rangle$ 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 *maxaug* 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_with_gaps* contact implementation is currently the only contact algorithm that supports friction. 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.
6. In a contact problem, FEBio calculates the projection of each slave node on the master surface. As a slave node slides across the master surface, the corresponding master segment can change. However, in some cases, switching segments 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, slave nodes will not be allowed to switch to new master segments. The default value is 0, which means that FEBio will do a segment update each iteration of each timestep.

7. The *sliding2* and *sliding3* 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 controls which version of the algorithm is used.

```
<symmetric_stiffness>0</symmetric_stiffness>
```

A value of 1 uses the symmetric version, where a value of 0 uses the non-symmetric version.

8. The *search_tol* parameter defines the search tolerance of the algorithm that projects slave nodes onto master facets. 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 master elements and due to numerical error may be projected outside the master surface.

3.7.3.2. Biphasic Contact

The *sliding2* implementation for sliding interfaces can deal with biphasic contact surfaces (including biphasic-on-biphasic, biphasic-on-elastic, biphasic-on-rigid). 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).

When performing biphasic-solute-on-rigid contact, a two-pass analysis should not be used; the biphasic-solute surface should be the slave surface.

3.7.3.3. Biphasic-Solute Contact

The *sliding3* 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). It allows for the possibility to track fluid and solute flow across the contact interface. 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>
```

In the same way that the *penalty* parameter controls the contact tractions, these 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, 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 contact, a two-pass analysis should not be used; the biphasic-solute surface should be the slave surface.

3.7.3.4. Rigid Wall Interfaces

A rigid wall interface is similar to a sliding interface, except that the master surface is a rigid wall. The following properties are defined for rigid wall interfaces:

Parameter	Description	Default
-----------	-------------	---------

tolerance	augmentation tolerance	0.01
penalty	penalty factor	1.0
plane	the plane equation for the rigid wall	N/A

The *plane* property defines the plane equation for the rigid wall. Its value is an array of four values: a, b, c, d_0 . It also 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 lc="1">0,0,1,0</plane>
```

The slave surface is defined by specifying a *surface* element. The *surface* tag is followed by the definition of the surface elements:

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

Triangular elements (tria3) may also be used instead of quadrilateral elements (quad4).

3.7.3.5. Tied Interfaces

A tied interface can be used to connect two non-conforming meshes. A tied interface requires the definition of both a slave and a master surface. It is assumed that the nodes of the slave surface are connected to the faces of the master surface. The following control parameters need to be defined:

<penalty>	penalty factor
<tolerance>	augmentation tolerance

The slave and master surfaces are defined similarly as for the sliding interfaces. The *type* attribute is used to specify whether it is a *slave* or *master* surface. The *surface* tag is followed by the definition of the surface elements:

```
<surface type="master">
  <quad4 id="n">n1,n2,n3,n4</quad4>
  ...
</surface>
```

Both quadrilateral surface elements (quad4) and triangular elements (tri3) may be used to define the surface.

3.7.3.6. Rigid Interfaces

A rigid interface defines a list of nodes that are attached to a rigid body. These nodes will move with the rigid body:

```
<contact type="rigid">
  <node id="n1" rb="1"></node>
  ...
  <node id="n2" rb="1"></node>
</contact>
```

The *id* attribute identifies the node and *rb* is the material id of the rigid body. The value of the node is ignored.

3.7.3.7. Rigid Joints

A rigid joint connects two rigid bodies at a point in space:

```
<contact type="rigid joint">
  <tolerance>0.1</tolerance>
  <penalty>2</penalty>
  <body1>1</body1>
  <body2>2</body2>
  <joint>0,0,0</joint>
</contact>
```

The *tolerance* element defines the augmentation tolerance. That is, when the relative change in the constraint forces (the Lagrange multipliers) are less than this value. The *body1* and *body2* elements are the material numbers of the two rigid bodies. The *joint* element defines the position of the joint 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.

3.7.4. Spring section

In FEBio you can connect two nodes via a discrete spring. The spring will exert a force on the nodes, depending on the separation distance and a spring constant.

A spring is defined using the `spring` element.

```
<spring [type="<type>"]>
```

The *type* attribute defines the type of spring. It can be any of the following values.

Type	Description
linear	spring that has a linear force-displacement relation

tension-only linear	like linear spring but force is only applied in tension
nonlinear	user defines the force-displacement relation

If the type attribute is omitted, it is assumed that the spring is linear.

All spring types must define the `node` sub-element to define which two nodes are connected through the spring.

The `linear` and `tension-only linear` elements require the `E` sub-element that defines the spring constant.

The `nonlinear` spring requires the `force` sub-element which defines the loadcurve that will be used for the force-displacement relation.

For example, you can define a linear spring between nodes 2 and 4 and having a spring constant of 2.5 as follows.

```
<spring type="linear">
  <node>2,4</node>
  <E>2.5</E>
</spring>
```

To define a nonlinear spring between nodes 2 and 4 use the following.

```
<spring type="nonlinear">
  <node>2,4</node>
  <force lc="1">1.0</force>
</spring>
```

In the last example, the force-displacement is defined by loadcurve 1. The value of the `force` element is the force scale factor, which in this case is one.

3.8. Loads Section

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

3.8.1. Nodal Loads

Nodal loads are applied by the *force* element. 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.

```
<force>
  <node id="n" bc="x" [lc="1"]>3.14</node>
  ...
</force>
```

The *id* attribute indicates to which node this prescribed dof is applied.

The *bc* attribute gives 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 3.10.2); for example, “*c2*”. An optional *loadcurve* can be specified with the *lc* attribute. If a loadcurve is not specified, the value will be automatically ramped from a value of 0 at time $t=0$ to the value specified in the xml file at the time corresponding to the end of the analysis.

The value of the *node* element (e.g. 3.14 in the example above) is the value for the nodal force. Note that if a *loadcurve* is specified, this value scales the value determined by the *loadcurve*.

3.8.2. Pressure forces

Pressure forces are applied to the surface of the geometry and are defined by the *pressure* element:

```
<pressure>
  <quad4 id="n" [lc="1" scale="1.0"]>n1,n2,n3,n4</quad4>
  ...
</pressure>
```

The sub-elements define the geometry of the pressure boundary surface. Each element must be either *quad4* for a four-node quadrilateral or *tri3* for a 3-noded triangular element. The id n references the surface element. 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 optional parameter *lc* defines a loadcurve for the pressure evolution and *scale* defines a scale factor. The default scale factor is 1.0 and if *lc* is not defined the scale factor is automatically ramped from 0 at time $t=0$ to the value specified here.

3.8.3. Mixture Normal Traction

This section applies to biphasic, biphasic-solute and triphasic 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):

```
<normal_traction traction="mixture">
  <quad4 id="n" [lc="1" scale="1.0"]>n1,n2,n3,n4</quad4>
  ...
</normal_traction>
```

or

```
<normal_traction traction="effective">
  <quad4 id="n" [lc="1" scale="1.0"]>n1,n2,n3,n4</quad4>
  ...
</normal_traction>
```

The optional parameter *lc* defines a loadcurve for the normal traction evolution and *scale* defines a scale factor. The default scale factor is 1.0, and if *lc* is not defined the scale factor is automatically ramped from 0 at time $t=0$ to the value specified here.

Unlike the mixture and effective traction, the fluid pressure p is a nodal variable (see Prescribed Nodal Degrees of Freedom). 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>
  <fix>
    <node id="n1" bc="p" lc="1">1.0</node>
    <node id="n2" bc="p" lc="1">1.0</node>
    <node id="n3" bc="p" lc="1">1.0</node>
    <node id="n4" bc="p" lc="1">1.0</node>
    ...
  </fix>
  <normal_traction traction="mixture">
    <quad4 id="n" lc="1" scale="-1.0" >n1,n2,n3,n4</quad4>
    ...
  </normal_traction>
</Boundary>
<LoadData>
  <loadcurve id="1">
    <loadpoint>0,0</loadpoint>
    <loadpoint>1,2.0</loadpoint>
  </loadcurve>
</LoadData>
```

3.8.4. 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:

```
<fluidflux type="nonlinear/nonlinear" flux="fluid">
  <quad4 id="n" [lc="1" scale="1.0"]>n1,n2,n3,n4</quad4>
  ...
</fluidflux>
```

The optional parameter lc defines a loadcurve for the normal flux evolution, and $scale$ defines a scale factor. The default scale factor is 1.0, and if lc is not defined the scale factor is automatically ramped from 0 at time $t=0$ to the value specified here.

type="nonlinear" (default) 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.

type="linear" 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

```
<fluidflux type="linear/nonlinear" flux="mixture">
  <quad4 id="n" [lc="1" scale="1.0"]>n1,n2,n3,n4</quad4>
  ...
</fluidflux>
```

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.

3.8.5. Solute Flux

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

```
<soluteflux type="linear/nonlinear" [sol="1"]>
  <quad4 id="n" [lc="1" scale="1.0"]>n1,n2,n3,n4</quad4>
  ...
</soluteflux>
```

The optional parameter *sol* specifies to which solute this flux condition applies, referencing the corresponding list in the Globals section (Section 3.10.2). If *sol* is not defined, the default value

is 1. The optional parameter *lc* defines a loadcurve for the normal flux evolution; *scale* defines a scale factor. The default scale factor is 1.0. If *lc* is not defined, the scale factor is automatically ramped from 0 at time $t=0$ to the value specified here.

type="nonlinear" (default) 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.

type="linear" 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), 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). 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.

3.8.6. Body Force

The body force 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.

3.8.6.1. Constant Body Force

This type of body force is spatially homogeneous, though it may vary with time when associated with a load curve:

```
<body_force type="const">
  <x lc="1">1.0</x>
  <y lc="2">1.0</y>
  <z lc="3">1.0</z>
</body_force>
```

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.

3.8.6.2. Non-Constant Body Force

This type of body force may be spatially inhomogeneous. The spatial inhomogeneity may be specified using a formula with variables x , y , z . For example:

```
<body_force type="non-const">
  <x>x+y+z</x>
  <y>x-y-z</y>
  <z>x*x+z</z>
</body_force>
```

3.8.6.3. Centrifugal Force

A centrifugal body force 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} .

```
<body_force 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_force>
```

3.9. Constraints Section

The Constraints section allows the user to enforce additional constraints to the model. Currently, only rigid body constraints can be enforced.

3.9.1. Rigid Body Constraints

Rigid bodies are initially unconstrained which means they can move in all three directions and can rotate about all three axes. To constrain the degrees of freedom of a rigid body you can use the *rigid_body* element:

```
<Constraints>
  <rigid_body>
    <!-- constraints go here -->
  </rigid_body>
</Constraints>
```

The following table lists the elements that can be defined in the *rigid_body* element:

Element	Description
<trans_x>	x-translation code. Attribute <i>type</i> specifies the type of the translation.
<trans_y>	y-translation code. Attribute <i>type</i> specifies the type of the translation.
<trans_z>	z-translation code. Attribute <i>type</i> specifies the type of the translation.
<rot_x>	x-rotation code. Attribute <i>type</i> specifies the type of the rotation.
<rot_y>	y-rotation code. Attribute <i>type</i> specifies the type of the rotation.
<rot_z>	z-rotation code. Attribute <i>type</i> specifies the type of the rotation.

The translation and rotation boundary codes are used to constrain the motion of the rigid body. The full syntax for e.g. the x-translation is as follows:

```
<trans_x type="<type>" [lc="<lc>"]> 1.0 </trans_x>
```

The *type* attribute can take on any of the following values:

Type	Description
fixed	Degree of freedom is fixed
prescribed	Degree of freedom is prescribed by user
force	A force is applied in direction of degree of freedom

If the type is *prescribed* or *force* then the *lc* attribute can be used to specify a load curve defining the amplitude of the displacement or force. The value is then interpreted as a scale factor. For all other types the value is ignored. The syntax and interpretation is the same for the other translation and rotation codes.

When the type is *force*, the force is applied at the center of mass for translational degrees of freedom and torque is applied around the center of mass for rotational degrees of freedom. The

center of mass of a rigid body is either specified in the material definition or calculated automatically by FEBio.

Example:

```
<rigid_body mat="1">
  <trans_x type="prescribed" lc="1">2.0</trans_x>
  <trans_y type="fixed"/>
  <trans_z type="fixed"/>
  <rot_x type="fixed"/>
  <rot_y type="fixed"/>
  <rot_z type="fixed"/>
</rigid_body>
```

In this example the rigid body that corresponds to material definition *1* has a prescribed displacement defined for the *x* degree of freedom and has all other degrees of freedom fixed.

A force (torque) can be applied at the center of mass by setting the *type* attribute to *force*. Note that specifying a force (torque) will automatically free the corresponding translational (rotational) degree of freedom. For example, applying a force in the *x*-direction while keeping the *y* and *z* directions fixed and the rotational degrees of freedom free, can be done as follows:

```
<rigid_body mat="1" >
  <trans_x type="force" lc="1">1.0</trans_x>
  <trans_y type="fixed"/>
  <trans_z type="fixed"/>
</rigid_body>
```

3.10. Globals Section

The *Globals* section is used to define some global variables, such as body forces and global constants.

3.10.1. Constants

Global constants currently include the universal gas constant R , absolute temperature θ , and Faraday constant F_c . 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>
```

3.10.2. 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 charge number:

```
<Globals>
  <Solutes>
    <solute id="1" name="Na">1</solute>
    <solute id="2" name="Cl">-1</solute>
    <solute id="3" name="Glc">0</solute>
  </Solutes>
</Globals>
```

In this example, the charge number of Na is +1, that of Cl is -1, and that of Glc is 0. These solute identification numbers should be referenced in the *sol* attribute of solutes when defining a biphasic-solute (Section 4.4.2), triphasic (Section 4.5.2) or multiphasic material.

3.11. LoadData Section

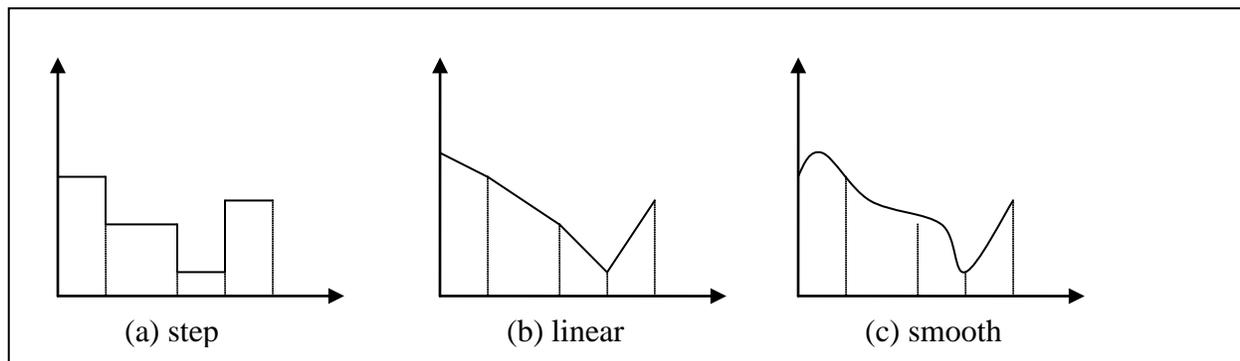
The *LoadData* section contains the loadcurve data. A loadcurve is defined by the *loadcurve* element. Each loadcurve is defined by repeating the *loadpoint* element for all data points:

```
<loadcurve id="1" [type="type" extend="extend"]>
  <loadpoint> 0, 0 </loadpoint>
  ...
  <loadpoint> 1, 1 </loadpoint>
</loadcurve>
```

The *id* attribute is the loadcurve number and is used in other sections of the input file as a means to reference this curve.

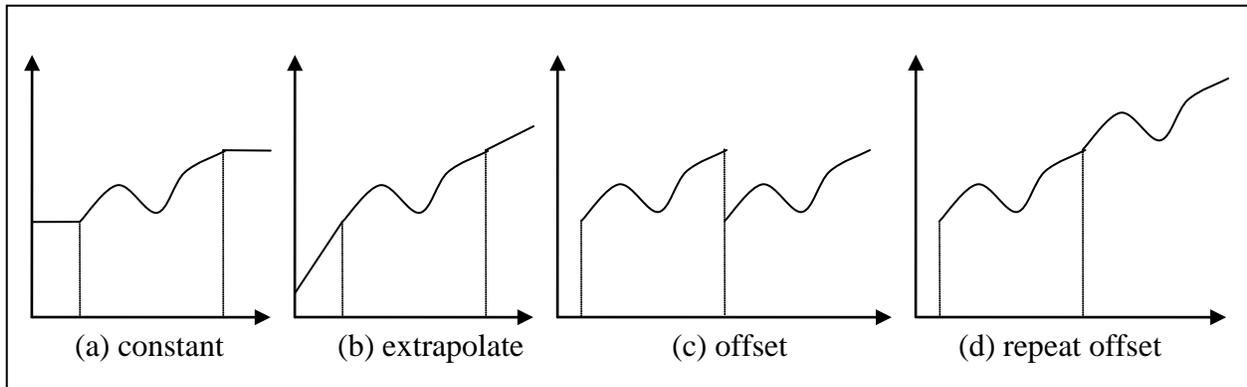
The optional attributes *type* and *extend* define how the value of the loadcurve is interpolated from the data points. The *type* defines the interpolation function and *extend* defines how the values of the loadcurve are determined outside of the interval defined by the first and last data point. The following tables list the possible values. The default values are marked with an asterisk (*).

Type	Description
step	Use a step interpolation function
linear*	Use a linear interpolation function
smooth	The values are interpolated using a cubic spline.



The different values for the *type* attribute 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* attribute of the load curve.

3.12. 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.

3.12.1. 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      [attribute list]>item list</node_data>
    <element_data   [attribute list]>item list</element_data>
    <rigid_body_data [attribute list]>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
- *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)

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. Note that the filename given is a template. FEBio appends a number at the end of the filename to indicate to which timestep the data belongs. For instance, if you define a file name as follows:

```
file = "data.txt"
```

then the first file that is written will have the name *data000.txt*. After the first converged timestep a file with name *data001.txt* will be written and so on.

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 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 *n*th 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.

3.12.1.1. 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
----	------------------------------------

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
cn	effective concentration of solute n

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>
```

3.12.1.2. 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
Fx	xx-component of the deformation gradient
Fy	yy-component of the deformation gradient
Fz	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
J	relative volume (determinant of deformation gradient)

For analyses using biphasic, biphasic-solute, and triphasic materials, the following additional variables can be defined:

Element variables	Description
p	actual fluid pressure
c_n	actual concentration of solute n
wx	x-component of fluid flux
wy	y-component of fluid flux
wz	z-component of fluid flux
j_{nx}	x-component of flux of solute n
j_{ny}	y-component of flux of solute n
j_{nz}	z-component of flux of solute n

For example, to store the (average) Cauchy stress for all elements, define the following data element:

```
<element_data data="sxx;sy;sz;sy;sz;sxz" name="element stresses">
</element_data>
```

3.12.1.3. 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
y	y-coordinate of center of mass
z	z-coordinate of center of mass

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

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>
```

3.12.2. 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). For backward compatibility, users can also save the results in the LSDYNA database format [1]. See Appendix B for information on how to use this format. This section describes how to customize the data that is stored in the *xplt* format.

To use this new format you need to define the *plotfile* element in the Output section of the FEBio input file.

```
<plotfile type="febio" [file="name.xplt"]/>
```

The *file* attribute is 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 *var* elements in the *plotfile* section.

```
<var type="name"/>
```

where *name* is the name of the variable. The following table lists the possible values for *name*.

Name	Description
acceleration	Nodal accelerations
anion concentration	Anion concentration
anion flux	Anion flux
cation concentration	Cation concentration

cation flux	Cation flux
contact gap	Contact gap distance
contact pressure	Contact pressures
contact traction	Contact traction vectors
current density	Current density
damage	Damage
displacement	Nodal displacements
effective anion concentration	Effective anion concentration
effective cation concentration	Effective cation concentration
effective fluid pressure	Effective fluid pressure
effective solute concentration	Effective solute concentration
electric potential	Electric potential
fixed charge density	Fixed charge density
fiber vector	Material fiber vector
fluid flux	Fluid flux
fluid pressure	Fluid pressure
reaction forces	Reaction force vectors
receptor-ligand concentration	Receptor-ligand concentration
relative volume	Relative volume
shell thickness	Shell thickness
solute concentration	Solute concentration
solute flux	Solute flux
stress	Cauchy stress
temperature	Nodal temperatures
velocity	Nodal velocities

The following example stores nodal displacements and element stresses.

Example:

```
<plotfile type="febio">
  <var type="displacement"/>
  <var type="stress"/>
</plotfile>
```

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.

4.1.1. Specifying fiber orientation

Some of the anisotropic materials are modeled as an isotropic matrix in which fibers are embedded. For these materials, the user must specify several parameters related to the fibers, including an initial fiber orientation. This orientation can be specified in several ways. FEBio gives the option to automatically generate the fiber orientation, based on some user-specified parameters. However, the user can override this feature and specify the fiber directions for each element manually in the *ElementData* section. See section 3.5.3 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.

<i>type</i> Value	Description
local	Use local element numbering (1)
spherical	Define the fiber orientation as a constant vector (2)
vector	specifies a spherical fiber distribution (3)
user	fibers are defined by users in the <i>ElementData</i> section. (4)

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>
```

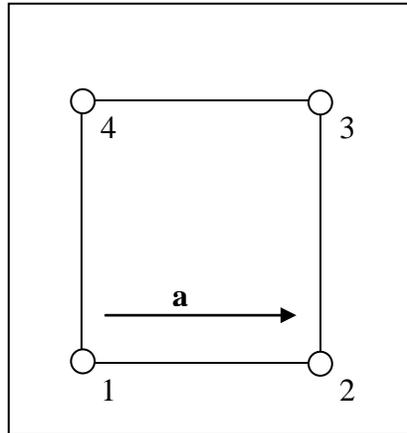


Figure 4-1. *local fiber direction option.*

- 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>
```

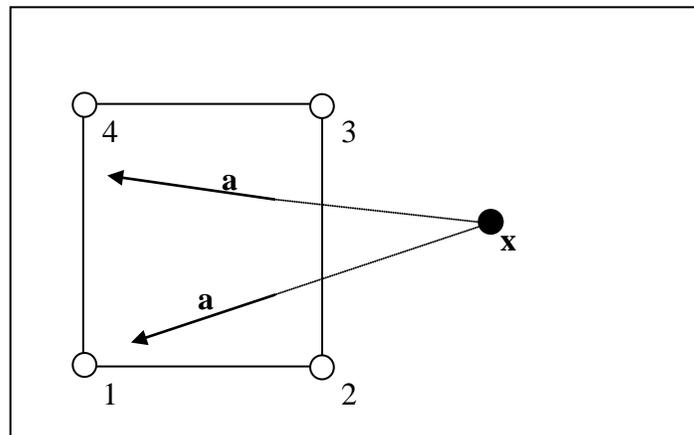


Figure 4-2. *spherical fiber direction option.*

- 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>
```

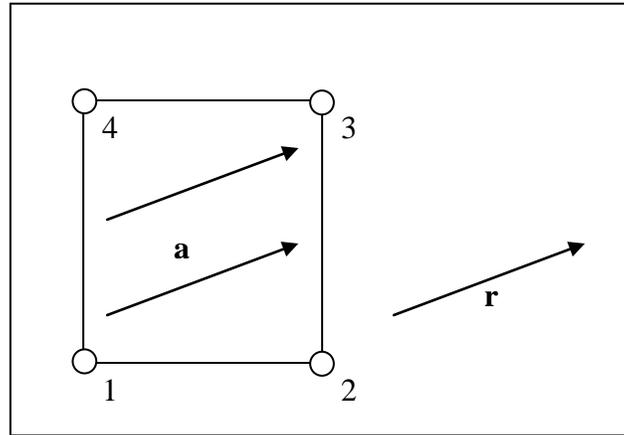


Figure 4-3. *vector* fiber direction option.

4. *user*: the fiber distribution is specified in the *ElementData* section. In this section you can define a different fiber direction for each element separately. See section 3.5 for more details. In this case the value is ignored. This option is useful when experimental data for fiber direction are available.

```
<fiber type="user"></fiber>
```

4.1.1.2. Orthotropic materials

For orthotropic materials, the user needs to specify two fiber directions **a** and **d**. From these FEBio will generate an orthonormal set of fiber vectors as follows:

$$\mathbf{e}_1 = \frac{\mathbf{a}}{\|\mathbf{a}\|}, \mathbf{e}_2 = \mathbf{e}_3 \times \mathbf{e}_1, \mathbf{e}_3 = \frac{\mathbf{a} \times \mathbf{d}}{\|\mathbf{a} \times \mathbf{d}\|}$$

The vectors **a** and **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 a and d directly. (2)

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>
```

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 [5]:

$$\Psi(\mathbf{C}) = \tilde{\Psi}(\tilde{\mathbf{C}}) + U(J),$$

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 2nd Piola-Kirchhoff stress is given by

$$\mathbf{S} = J^{-2/3} \text{Dev}[\tilde{\mathbf{S}}] + pJ\mathbf{C}^{-1},$$

where

$$\tilde{\mathbf{S}} = 2 \frac{\partial \tilde{\Psi}}{\partial \tilde{\mathbf{C}}},$$

and

$$p := \frac{dU}{dJ},$$

and $\text{Dev}[\cdot]$ is the deviatoric operator in the material frame.

The corresponding Cauchy stress is given by

$$\boldsymbol{\sigma} = \text{dev}[\tilde{\boldsymbol{\sigma}}] + p\mathbf{I},$$

where $\tilde{\boldsymbol{\sigma}} = J^{-1} \tilde{\mathbf{F}} \cdot \tilde{\mathbf{S}} \cdot \tilde{\mathbf{F}}^T$ and $\text{dev}[\cdot]$ is the deviatoric operator in the spatial frame.

For these materials, the entire bulk (volumetric) behavior is determined by the function $U(J)$, and p represents the entire hydrostatic stress. The function $U(J)$ 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 [5]. 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 I (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 [6]. The following material parameters are required:

mu	initial modulus
N	number of links in chain
k	Bulk modulus

The uncoupled strain energy function for the Arruda-Boyce material is given by:

$$\Psi = \mu \sum_{i=1}^5 \frac{C_i}{N^{i-1}} (\tilde{I}_1^i - 3^i) + U(J),$$

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 [6] 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

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 compressible material that acts as a ground matrix, using a “*uncoupled solid mixture*” container as described in Section 4.1.2.14. . The following material parameters need to be defined:

<beta>	parameters ($\beta_1, \beta_2, \beta_3$)
<ksi>	parameters (ξ_1, ξ_2, ξ_3)

The stress $\tilde{\boldsymbol{\sigma}}$ for this material is given by [7-9]:

$$\tilde{\boldsymbol{\sigma}} = \int_0^{2\pi} \int_0^\pi H(\tilde{I}_n - 1) \tilde{\boldsymbol{\sigma}}_n(\mathbf{n}) \sin \varphi d\varphi d\theta.$$

$\tilde{I}_n = \tilde{\lambda}_n^2 = \mathbf{N} \cdot \tilde{\mathbf{C}} \cdot \mathbf{N}$ 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 $H(\cdot)$ 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,

$$\tilde{\boldsymbol{\sigma}}_n = \frac{2\tilde{I}_n}{J} \frac{\partial \tilde{\Psi}}{\partial \tilde{I}_n} \mathbf{n} \otimes \mathbf{n},$$

where in this material,

$$\tilde{\Psi}(\mathbf{n}, \tilde{I}_n) = \xi(\mathbf{n}) (\tilde{I}_n - 1)^{\beta(\mathbf{n})}.$$

The materials parameters β and ξ are determined from:

$$\xi(\mathbf{n}) = \left(\frac{\cos^2 \theta \sin^2 \varphi}{\xi_1^2} + \frac{\sin^2 \theta \sin^2 \varphi}{\xi_2^2} + \frac{\cos^2 \varphi}{\xi_3^2} \right)^{-1/2}$$

$$\beta(\mathbf{n}) = \left(\frac{\cos^2 \theta \sin^2 \varphi}{\beta_1^2} + \frac{\sin^2 \theta \sin^2 \varphi}{\beta_2^2} + \frac{\cos^2 \varphi}{\beta_3^2} \right)^{-1/2}.$$

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
<c2>	Mooney-Rivlin parameter c2
<k>	bulk modulus
<beta>	parameters $(\beta_1, \beta_2, \beta_3)$
<ksi>	parameters (ξ_1, ξ_2, ξ_3)

The stress $\tilde{\sigma}$ for this material is given by,

$$\tilde{\sigma} = \tilde{\sigma}_{MR} + \tilde{\sigma}_f.$$

Here, $\tilde{\sigma}_{MR}$ is the stress from the Mooney-Rivlin basis (Section 4.1.2.7.), and $\tilde{\sigma}_f$ 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
<c2>	Second VW coefficient
<k>	Bulk modulus
<beta>	parameters $(\beta_1, \beta_2, \beta_3)$
<ksi>	parameters (ξ_1, ξ_2, ξ_3)

The stress $\tilde{\sigma}$ for this material is given by,

$$\tilde{\sigma} = \tilde{\sigma}_{vw} + \tilde{\sigma}_f.$$

Here, $\tilde{\sigma}_{vw}$ is the stress from the Veronda-Westmann basis (Section 4.1.2.15.), and $\tilde{\sigma}_f$ is the stress contribution from the ellipsoidal fiber distribution (Section 4.1.2.3.).

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. 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 compressible material that acts as a ground matrix, using a “*uncoupled solid mixture*” container as described in Section 4.1.2.14. . The following material parameters need to be defined:

<ksi>	ξ , representing a measure of the fiber modulus
<alpha>	α , coefficient of exponential argument
<beta>	β , power of exponential argument
<theta>	θ , spherical angle for fiber orientation in local coordinate system (in degrees)
<phi>	φ , spherical angle for fiber orientation in local coordinate system (in degrees)

The fiber is oriented along

$$\mathbf{N} = \sin \varphi \cos \theta \mathbf{e}_1 + \sin \varphi \sin \theta \mathbf{e}_2 + \cos \varphi \mathbf{e}_3,$$

where $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ are orthonormal vectors representing the local element coordinate system (Section 4.1.1). The stress $\tilde{\boldsymbol{\sigma}}$ for this fibrous material is given by

$$\tilde{\boldsymbol{\sigma}} = H(\tilde{I}_n - 1) \frac{2\tilde{I}_n}{J} \frac{\partial \tilde{\Psi}}{\partial \tilde{I}_n} \mathbf{n} \otimes \mathbf{n},$$

where $\tilde{I}_n = \tilde{\lambda}_n^2 = \mathbf{N} \cdot \tilde{\mathbf{C}} \cdot \mathbf{N}$ is the square of the fiber stretch, $\mathbf{n} = \tilde{\mathbf{F}} \cdot \mathbf{N} / \tilde{\lambda}_n$, and $H(\cdot)$ is the unit step function that enforces the tension-only contribution.. The fiber strain energy density is given by

$$\tilde{\Psi} = \frac{\xi}{\alpha\beta} \left(\exp \left[\alpha (\tilde{I}_n - 1)^\beta \right] - 1 \right),$$

where $\xi > 0$, $\alpha \geq 0$, and $\beta \geq 2$.

Note: In the limit when $\alpha \rightarrow 0$, this expressions produces a power law,

$$\lim_{\alpha \rightarrow 0} \tilde{\Psi} = \frac{\xi}{\beta} (\tilde{I}_n - 1)^\beta.$$

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 neo-Hookean ground matrix.

```
<material id="1" type="uncoupled 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>
    <theta>0</theta>
    <phi>90</phi>
  </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="uncoupled 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>
    <theta>90</theta>
    <phi>25</phi>
  </solid>
  <solid type="fiber-exp-pow">
    <ksi>5</ksi>
    <alpha>20</alpha>
    <beta>3</beta>
    <theta>-90</theta>
    <phi>25</phi>
  </solid>
</material>

```

4.1.2.6. Fung Orthotropic

The material type for orthotropic Fung elasticity [10, 11] is “*Fung orthotropic*”. The following material parameters must be defined:

<mu1>	μ_1 modulus
<mu2>	μ_2 modulus
<mu3>	μ_3 modulus
<l11>	λ_{11} modulus
<l22>	λ_{22} modulus
<l33>	λ_{33} modulus
<l12>	λ_{12} modulus
<l23>	λ_{23} modulus
<l31>	λ_{31} modulus
<c>	c coefficient
<k>	bulk modulus

The hyperelastic strain energy function is given by [12],

$$\Psi = \frac{1}{2}c(e^{\tilde{Q}} - 1) + U(J), \quad (0.1)$$

where,

$$\tilde{Q} = c^{-1} \sum_{a=1}^3 \left[2\mu_a \mathbf{M}_a : \tilde{\mathbf{E}}^2 + \sum_{b=1}^3 \lambda_{ab} (\mathbf{M}_a : \tilde{\mathbf{E}})(\mathbf{M}_b : \tilde{\mathbf{E}}) \right].$$

Here, $\tilde{\mathbf{E}} = (\tilde{\mathbf{C}} - \mathbf{I})/2$ 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 orthotropic Lamé parameters should produce a positive definite stiffness matrix.

Example:

```
<material id="3" type="Fung orthotropic">
  <mu1>1</mu1>
  <mu2>2</mu2>
  <mu3>3</mu3>
  <l11>3</l11>
  <l22>2</l22>
  <l33>1</l33>
  <l12>0.5</l12>
  <l23>1.0</l23>
  <l31>1.5</l31>
  <c>1</c>
  <K>100</K>
</material>
```

4.1.2.7. 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
<c2>	Coefficient of second invariant term
<k>	Bulk modulus

This material model is a hyperelastic Mooney-Rivlin type with uncoupled deviatoric and volumetric behavior. The strain-energy function is given by:

$$\Psi = C_1(\tilde{I}_1 - 3) + C_2(\tilde{I}_2 - 3) + \frac{1}{2}K(\ln J)^2.$$

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 [5].

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>
```

4.1.2.8. Muscle Material

This material model implements the constitutive model developed by Silvia S. Blemker [13]. 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
<g2>	cross fiber shear modulus
<p1>	exponential stress coefficients
<p2>	fiber uncrimping factor
<Lofl>	optimal fiber length
<smax>	maximum isometric stress
<lambda>	fiber stretch for straightened fibers
<k>	bulk modulus
<active_contraction>	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 [14], instead of the usual five invariants proposed by A.J.M. Spencer [15]. For this particular material, only two of the five Criscione invariants are used. The strain energy function is defined as follows:

$$\Psi(B_1, B_2, \lambda) = G_1 \tilde{B}_1^2 + G_2 \tilde{B}_2^2 + F_m(\tilde{\lambda}) + U(J).$$

The function F_m is the strain energy contribution of the muscle fibers. It is defined as follows:

$$\lambda \frac{\partial F_m}{\partial \lambda} = \sigma_{\max} \left(f_m^{\text{passive}}(\lambda) + \alpha f_m^{\text{active}}(\lambda) \right) \frac{\lambda}{\lambda_{\text{ofl}}},$$

where,

$$f_m^{\text{passive}}(\lambda) = \begin{cases} 0 & , \lambda \leq \lambda_{\text{ofl}} \\ P_1 \left(e^{P_2(\lambda/\lambda_{\text{ofl}} - 1)} - 1 \right) & , \lambda_{\text{ofl}} < \lambda < \lambda^* \\ P_3 \lambda / \lambda_{\text{ofl}} + P_4 & , \lambda \geq \lambda^* \end{cases},$$

and,

$$f_m^{\text{active}}(\lambda) = \begin{cases} 9 \left(\lambda / \lambda_{\text{ofl}} - 0.4 \right)^2 & , \lambda \leq 0.6 \lambda_{\text{ofl}} \\ 9 \left(\lambda / \lambda_{\text{ofl}} - 1.6 \right)^2 & , \lambda \geq 1.4 \lambda_{\text{ofl}} \\ 1 - 4 \left(1 - \lambda / \lambda_{\text{ofl}} \right)^2 & , 0.6 \lambda_{\text{ofl}} < \lambda < 1.4 \lambda_{\text{ofl}} \end{cases},$$

The values P_3 and P_4 are determined by requiring C^0 and C^1 continuity at $\lambda = \lambda^*$.

The parameter α is the activation level and can be specified using the *active_contraction* 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>  
  <Lof1>1.07</Lof1>  
  <lambda>1.4</lambda>  
  <k>1e6</k>  
  <fiber type="vector">1,0,0</fiber>  
</material>
```

4.1.2.9. Ogden

This material describes an incompressible hyperelastic Ogden material [5]. The following material parameters must be defined:

<c[n]>	Coefficient of n th term, where n can range from 1 to 6
<m[n]>	Exponent of n th term, where n can range from 1 to 6
<k>	Bulk modulus

The uncoupled hyperelastic strain energy function for this material is given in terms of the eigenvalues of the deformation tensor:

$$\Psi = \sum_{i=1}^N \frac{c_i}{m_i^2} (\tilde{\lambda}_1^{m_i} + \tilde{\lambda}_2^{m_i} + \tilde{\lambda}_3^{m_i} - 3) + U(J).$$

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>
```

4.1.2.10. Tendon Material

The material type for the tendon material is *tendon material*. The tendon material is similar to the muscle material. The only difference is the fiber function. For tendon material this is defined as:

$$\lambda \frac{\partial F_t}{\partial \lambda} = \sigma(\lambda),$$

where

$$\sigma(\lambda) = \begin{cases} 0, & \lambda \leq 1 \\ L_1 \left(e^{L_2(\lambda-1)} - 1 \right) & 1 < \lambda < \lambda^* \\ L_3 \lambda + L_4 & \lambda \geq \lambda^* \end{cases}.$$

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
<g2>	cross fiber shear modulus
<l1>	exponential stress coefficients
<l2>	fiber uncrimping factor
<lambda>	fiber stretch for straightened fibers
<k>	bulk modulus

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>
```

4.1.2.11. 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
<c2>	Second Mooney-Rivlin material parameter
<k>	bulk modulus
<beta>	the β parameter (see below)
<ksi>	the ξ parameter (see below)
<mat_axis>	defines the material axes

This material is based on the following uncoupled hyperelastic strain energy function [16]:

$$\Psi(\mathbf{C}, \lambda_1, \lambda_2, \lambda_3) = \tilde{\Psi}_{iso}(\tilde{\mathbf{C}}) + \sum_{i=1}^3 \tilde{\Psi}_i^{TC}(\tilde{\lambda}_i) + U(J).$$

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:

$$\tilde{\Psi}_i^{TC}(\tilde{\lambda}_i) = \begin{cases} \xi_i (\tilde{\lambda}_i - 1)^{\beta_i}, & \tilde{\lambda}_i > 1 \\ 0, & \tilde{\lambda}_i \leq 1 \end{cases} \quad \xi_i \geq 0 \quad (\text{no sum on } i)$$

The $\tilde{\lambda}_i$ parameters are the deviatoric fiber stretches of the local material fibers:

$$\tilde{\lambda}_i = (\mathbf{a}_i^0 \cdot \tilde{\mathbf{C}} \cdot \mathbf{a}_i^0)^{1/2}.$$

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>
```

4.1.2.12. 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
<c2>	Mooney-Rivlin coefficient 2
<c3>	Exponential stress coefficient
<c4>	Fiber uncrimping coefficient
<c5>	Modulus of straightened fibers
<k>	Bulk modulus
<lam_max>	Fiber stretch for straightened fibers
<fiber>	Fiber distribution option

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 [17-19]. 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:

$$\Psi = F_1(\tilde{I}_1, \tilde{I}_2) + F_2(\tilde{\lambda}) + \frac{K}{2} [\ln(J)]^2.$$

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 $J = \det(\mathbf{F})$ 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:

$$\begin{aligned} \tilde{\lambda} \frac{\partial F_2}{\partial \tilde{\lambda}} &= 0, \quad \tilde{\lambda} \leq 1 \\ \tilde{\lambda} \frac{\partial F_2}{\partial \tilde{\lambda}} &= C_3 \left(e^{C_4(\tilde{\lambda}-1)} - 1 \right), \quad 1 < \tilde{\lambda} < \lambda_m. \\ \tilde{\lambda} \frac{\partial F_2}{\partial \tilde{\lambda}} &= C_5 + C_6 \tilde{\lambda}, \quad \tilde{\lambda} \geq \lambda_m \end{aligned}$$

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_3 scales 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 [5].

The fiber orientation can be specified as explained in Section 4.1.1. Active stress along the fiber direction can be simulated using an active contraction model. To use this feature you need to define the *active_contraction* parameter. This parameter takes an optional attribute, *lc*, which defines the loadcurve. There are also several options:

<ca0>	Intracellular calcium concentration
<beta>	tension-sarcomere length relation constant
<l0>	Unloaded sarcomere length
<refl>	No tension sarcomere length

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 lc="1">
    <ca0>4.35</ca0>
    <beta>4.75</beta>
    <l0>1.58</l0>
    <refl>2.04</refl>
  </active_contraction>
</material>
```

4.1.2.13. Transversely Isotropic Veronda-Westmann

The material type for transversely isotropic Veronda-Westmann materials is “*trans iso Veronda-Westmann*”. The following material parameters must be defined:

<c1>	Veronda-Westmann coefficient 1
<c2>	Veronda-Westmann coefficient 2
<c3>	Exponential stress coefficient
<c4>	Fiber uncrimping coefficient
<c5>	Modulus of straightened fibers
<k>	Bulk modulus
<lam_max>	Fiber stretch for straightened fibers
<fiber>	Fiber distribution option.

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 active contraction model can also be defined for this material. See the transversely isotropic Mooney-Rivlin model 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>
```

4.1.2.14. 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 compressible material
<k>	Bulk modulus

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. If a bulk modulus is specified for the <solid> material, it will be added to the bulk modulus of the mixture.

Example:

```
<material id="1" type="uncoupled solid mixture">
  <k>1000</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">
    <ksi>10, 12, 15</ksi>
    <beta>2.5, 3, 3</beta>
  </solid>
</material>
```

4.1.2.15. Veronda-Westmann

The material type for incompressible Veronda-Westmann materials is *Veronda-Westmann* [20]. The following material parameters must be defined:

<c1>	First VW coefficient
<c2>	Second VW coefficient
<k>	Bulk modulus

This model is similar to the Mooney-Rivlin model in that it also uses an uncoupled deviatoric dilatational strain energy:

$$\Psi = C_1 \left[e^{(C_2(\tilde{I}_1-3))} - 1 \right] - \frac{C_1 C_2}{2} (\tilde{I}_2 - 3) + U(J).$$

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 [20].

Example:

```
<material id="2" type="Veronda-Westmann">
  <c1>1000.0</c1>
  <c2>2000.0</c2>
  <k>1000</k>
</material>
```

4.1.3. Compressible 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 formulation. They should not be used for nearly-incompressible material behavior due to the potential for element locking.

4.1.3.1. Cell Growth

The material type for cell growth is “*cell growth*”. 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
<ce>	extracellular osmolarity, c_e

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>
```

4.1.3.2. 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 [21, 22]. 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 4.1.3.13. . The following material parameters need to be defined:

$\langle \phi_0^w \rangle$	gel porosity (fluid volume fraction) in reference (strain-free) configuration, ϕ_0^w
$\langle c_0^F \rangle$	fixed-charge density in reference (strain-free) configuration, c_0^F
$\langle c^* \rangle$	external bath osmolarity, \bar{c}^*

The Cauchy stress for this material is the stress from the Donnan equilibrium response [8]:

$$\boldsymbol{\sigma} = -\pi \mathbf{I},$$

where π is the osmotic pressure, given by

$$\pi = RT \left(\sqrt{(c^F)^2 + (\bar{c}^*)^2} - \bar{c}^* \right),$$

and c^F is the fixed-charge density in the current configuration, related to the reference configuration via

$$c^F = \frac{\phi_0^w}{J - 1 + \phi_0^w} c_0^F,$$

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.

Note that c_0^F may be negative or positive; the gel porosity is unitless and must be in the range $0 < \phi_0^w < 1$. 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 proelastic material as described in Section 4.3 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 $\bar{c}^* \rightarrow \infty$. 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>
```

4.1.3.3. 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 4.1.3.13. . The following material parameters need to be defined:

<beta>	parameters $(\beta_1, \beta_2, \beta_3)$
<ksi>	parameters (ξ_1, ξ_2, ξ_3)

The Cauchy stress for this fibrous material is given by [7-9]:

$$\boldsymbol{\sigma} = \int_0^{2\pi} \int_0^\pi H(I_n - 1) \boldsymbol{\sigma}_n(\mathbf{n}) \sin \varphi d\varphi d\theta .$$

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 $H(\cdot)$ is the unit step function that enforces the tension-only contribution.

The fiber stress is determined from a fiber strain energy function,

$$\boldsymbol{\sigma}_n = \frac{2I_n}{J} \frac{\partial \Psi}{\partial I_n} \mathbf{n} \otimes \mathbf{n} ,$$

where in this material,

$$\Psi(\mathbf{n}, I_n) = \xi(\mathbf{n})(I_n - 1)^{\beta(\mathbf{n})} .$$

The materials parameters β and ξ are assumed to vary ellipsoidally with \mathbf{n} , according to

$$\xi(\mathbf{n}) = \left(\frac{\cos^2 \theta \sin^2 \varphi}{\xi_1^2} + \frac{\sin^2 \theta \sin^2 \varphi}{\xi_2^2} + \frac{\cos^2 \varphi}{\xi_3^2} \right)^{-1/2}$$

$$\beta(\mathbf{n}) = \left(\frac{\cos^2 \theta \sin^2 \varphi}{\beta_1^2} + \frac{\sin^2 \theta \sin^2 \varphi}{\beta_2^2} + \frac{\cos^2 \varphi}{\beta_3^2} \right)^{-1/2} .$$

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>
```

4.1.3.4. 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
<v>	Poisson’s ratio
<beta>	parameters $(\beta_1, \beta_2, \beta_3)$
<ksi>	parameters (ξ_1, ξ_2, ξ_3)

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 4.1.3.10.), and $\boldsymbol{\sigma}_f$ is the stress contribution from the fibers (Section 4.1.3.3.).

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>
```

4.1.3.5. 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
<bosm>	external bath osmolarity, \bar{c}^*
<beta>	parameters $(\beta_1, \beta_2, \beta_3)$
<ksi>	parameters (ξ_1, ξ_2, ξ_3)

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 4.1.3.2.

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>
```

4.1.3.6. 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 4.1.3.13. . The following material parameters need to be defined:

<ksi>	ξ , representing a measure of the fiber modulus
<alpha>	α , coefficient of exponential argument
<beta>	β , power of exponential argument
<theta>	θ , spherical angle for fiber orientation in local coordinate system (in degrees)
<phi>	φ , spherical angle for fiber orientation in local coordinate system (in degrees)

The fiber is oriented along

$$\mathbf{N} = \sin \varphi \cos \theta \mathbf{e}_1 + \sin \varphi \sin \theta \mathbf{e}_2 + \cos \varphi \mathbf{e}_3, \quad 0 \leq \theta < 2\pi, 0 \leq \varphi \leq \pi,$$

where $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ are orthonormal vectors representing the local element coordinate system (Section 4.1.1). The Cauchy stress for this fibrous material is given by

$$\boldsymbol{\sigma} = H(I_n - 1) \frac{2I_n}{J} \frac{\partial \Psi}{\partial I_n} \mathbf{n} \otimes \mathbf{n},$$

where $I_n = \lambda_n^2 = \mathbf{N} \cdot \mathbf{C} \cdot \mathbf{N}$ is the square of the fiber stretch, $\mathbf{n} = \mathbf{F} \cdot \mathbf{N} / \lambda_n$, and $H(\cdot)$ is the unit step function that enforces the tension-only contribution.. The fiber strain energy density is given by

$$\Psi = \frac{\xi}{\alpha\beta} \left(\exp \left[\alpha (I_n - 1)^\beta \right] - 1 \right),$$

where $\xi > 0$, $\alpha \geq 0$, and $\beta \geq 2$.

Note: In the limit when $\alpha \rightarrow 0$, this expressions produces a power law,

$$\lim_{\alpha \rightarrow 0} \Psi = \frac{\xi}{\beta} (I_n - 1)^\beta$$

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:

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>
        <theta>0</theta>
        <phi>90</phi>
    </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>
        <theta>90</theta>
        <phi>25</phi>
    </solid>
    <solid type="fiber-exp-pow">
        <ksi>5</ksi>
        <alpha>20</alpha>
        <beta>3</beta>
        <theta>-90</theta>
        <phi>25</phi>
    </solid>
</material>

```

4.1.3.7. Holmes-Mow

The material type for the Holmes-Mow material [23] is *Holmes-Mow*. This isotropic hyperelastic material has been used to represent the solid matrix of articular cartilage [23, 24] and intervertebral disc [25]. The following material parameters must be defined:

<E>	Young's modulus
<v>	Poisson's ratio
<beta>	Exponential stiffening coefficient

The coupled hyperelastic strain-energy function for this material is given by [23]:

$$W(I_1, I_2, J) = \frac{1}{2}c(e^Q - 1),$$

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,

$$Q = \frac{\beta}{\lambda + 2\mu} \left[(2\mu - \lambda)(I_1 - 3) + \lambda(I_2 - 3) - (\lambda + 2\mu) \ln J^2 \right],$$

$$c = \frac{\lambda + 2\mu}{2\beta},$$

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:

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

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

Example:

```
<material id="3" type="Holmes-Mow">
  <E>1</E>
  <v>0.35</v>
  <beta>0.25</beta>
</material>
```

4.1.3.8. Isotropic Elastic

The material type for isotropic elasticity is *isotropic elastic*[§]. The following material parameters must be defined:

<E>	Young's modulus
<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:

$$W = \frac{1}{2} \lambda (\text{tr} \mathbf{E})^2 + \mu \mathbf{E} : \mathbf{E}.$$

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:

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{2(1+\nu)}.$$

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 4.1.3.7. , the neo-Hookean material in Section 4.1.3.10. , or the unconstrained Ogden material in Section 4.1.3.11. .

Example:

```
<material id="1" type="isotropic elastic"
  <E>1000.0</E>
  <v>0.45</v>
</material>
```

[§] 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.

4.1.3.9. 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
<E2>	Young’s modulus in the y-direction
<E3>	Young’s modulus in the z-direction
<G12>	Shear modulus in the xy-plane
<G23>	Shear modulus in the yz-plane
<G31>	Shear modulus in the xz-plane
<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>
```

4.1.3.10. Neo-Hookean

The material type for a Neo-Hookean material is *neo-Hookean*. The following parameters must be defined:

<E>	Young's modulus
<v>	Poisson's ratio

This model describes a compressible Neo-Hookean material [26]. 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:

$$W = \frac{\mu}{2}(I_1 - 3) - \mu \ln J + \frac{\lambda}{2}(\ln J)^2.$$

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 this case, use the *Mooney-Rivlin* material described in Section 4.1.2.7.

Example:

```
<material id="1" type="neo-Hookean">
  <E>1000.0</E>
  <v>0.45</v>
</material>
```

4.1.3.11. Ogden Unconstrained

This material describes a compressible (unconstrained) hyperelastic Ogden material [5]. The following material parameters must be defined:

<c[n]>	Coefficient of n th term, where n can range from 1 to 6
<m[n]>	Exponent of n th term, where n can range from 1 to 6
<cp>	Bulk-like modulus

The hyperelastic strain energy function for this material is given in terms of the eigenvalues of the right or left stretch tensor:

$$W(\lambda_1, \lambda_2, \lambda_3) = \frac{1}{2} c_p (J - 1)^2 + \sum_{i=1}^N \frac{c_i}{m_i^2} (\lambda_1^{m_i} + \lambda_2^{m_i} + \lambda_3^{m_i} - 3 - m_i \ln J).$$

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>
```

4.1.3.12. 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 4.1.3.13. . 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
<bosm>	external bath osmolarity, \bar{c}^*

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

$$\pi = RT(\bar{c} - \bar{c}^*).$$

\bar{c} is the interstitial fluid in the current configuration, related to the reference configuration via

$$\bar{c} = \frac{\phi_0^w}{J - 1 + \phi_0^w} \bar{c}_0$$

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 poroelastic material as described in Section 4.3 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>
```

4.1.3.13. Solid Mixture

This material describes a mixture of compressible elastic solids. It is a container for any combination of the materials described in Section 4.1.3.

<solid>	Container tag for compressible 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.

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">
    <ksi>10, 12, 15</ksi>
    <beta>2.5, 3, 3</beta>
  </solid>
</material>
```

4.1.3.14. 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 4.1.3.13. . The following material parameters need to be defined:

<beta>	parameter β
<ksi>	parameters ξ

The Cauchy stress for this fibrous material is given by [7-9]:

$$\boldsymbol{\sigma} = \int_0^{2\pi} \int_0^\pi H(I_n - 1) \boldsymbol{\sigma}_n(\mathbf{n}) \sin \varphi d\varphi d\theta.$$

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 $H(\cdot)$ is the unit step function that enforces the tension-only contribution.

The fiber stress is determined from a fiber strain energy function,

$$\boldsymbol{\sigma}_n = \frac{2I_n}{J} \frac{\partial \Psi}{\partial I_n} \mathbf{n} \otimes \mathbf{n},$$

where in this material,

$$\Psi(\mathbf{n}, I_n) = \xi (I_n - 1)^\beta.$$

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>
    <beta>2.5</beta>
  </solid>
</material>
```

4.1.3.15. Coupled Transversely Isotropic Mooney-Rivlin

This material describes a transversely isotropic Mooney-Rivlin material using a fully-coupled formulation. It is defined through the *coupled trans-iso Mooney-Rivlin* material type. The following material parameters must be defined.

c1	Mooney-Rivlin c_1 parameter.
c2	Mooney-Rivlin c_2 parameter.
c3	exponential multiplier
c4	fiber scale factor
c5	fiber modulus in linear region
lam_max	maximum fiber straightening stretch
k	bulk-like modulus

The strain-energy function for this constitutive model is defined by

$$W = c_1(I_1 - 3) + c_2(I_2 - 3) + F(\lambda) + U(J)$$

The first two terms define the 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 {Weiss, 1996 #14}. 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>
```

4.1.3.16. 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.
c2	Veronda-Westmann c_2 parameter.
c3	exponential multiplier
c4	fiber scale factor
c5	fiber modulus in linear region
lam_max	maximum fiber straightening stretch
k	bulk-like modulus

The strain-energy function for this constitutive model is defined by

$$W = c_1 \left[\exp(c_2 (I_1 - 3)) \right] - \frac{1}{2} c_1 c_2 (I_2 - 3) + F(\lambda) + U(J)$$

The first two terms define the 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 {Weiss, 1996 #14}. 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 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>
```

4.2. Viscoelastic Solids

4.2.1. Uncoupled Viscoelastic Materials

These materials will use the additive decomposition of the deformation gradient to separate volumetric and deviatoric material response. With this decomposition, it will be possible to produce viscoelastic volumetric behavior, viscoelastic deviatoric behavior, or both types of behavior simultaneously.

4.2.2. Compressible Viscoelastic Materials

The material type for viscoelastic materials is “*viscoelastic*”. The following parameters need to be defined:

<t1>-<t6>	relaxation times
<g1>-<g6>	viscoelastic coefficients
<solid_id>	material ID for the elastic component

For a viscoelastic material, the second Piola Kirchhoff stress can be written as follows [17]:

$$\mathbf{S}(t) = \int_{-\infty}^t G(t-s) \frac{d\mathbf{S}^e}{ds} ds,$$

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:

$$G(t) = 1 + \sum_{i=1}^N \gamma_i \exp(-t / \tau_i),$$

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 γ_2 have to be entered.

The *solid_id* parameter is the material ID of the elastic material. The elastic material is defined as a separate material in the FEBio input file. This allows the user to use any of the other hyperelastic materials as the elastic part of the viscoelastic material.

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>
```

4.3. 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. Therefore, the constitutive relation of the solid matrix should be chosen from the list of compressible materials provided in Section 4.1.3. 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

$$\mathbf{w} = -\mathbf{k} \cdot \nabla p$$

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.)

4.3.1. General Specification of Biphase Materials

The material type for a biphase material is “*biphase*”. The following parameters must be defined:

<solid>	Specification of the solid matrix
<phi0>	solid volume fraction φ_r^s in the reference configuration ($0 < \varphi_r^s < 1$)
<permeability>	Specification of the hydraulic permeability

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 4.1.3. 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 4.3.2. 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 $1 - \varphi_r^s$.

Example:

```
<material id="1" name="Biphase tissue" type="biphase">
  <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>
```

4.3.2. Permeability Materials

Permeability materials provide a constitutive relation for the hydraulic permeability of a biphasic material.

4.3.2.1. Constant Isotropic Permeability

The material type for constant isotropic permeability is “*const-iso-perm*”. The following material parameters must be defined:

<perm>	hydraulic permeability k
--------	----------------------------

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 [27, 28].

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="const-iso-perm">
  <perm>0.001</perm>
</permeability>
```

4.3.2.2. 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
<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 [23]:

$$\mathbf{k} = k(J)\mathbf{I},$$

where,

$$k(J) = k_0 \left(\frac{J - \varphi_0}{1 - \varphi_0} \right)^\alpha e^{\frac{1}{2}M(J^2 - 1)},$$

and J is the Jacobian of the deformation, i.e. $J = \det \mathbf{F}$ where \mathbf{F} is the deformation gradient.

Example:

This example defines a poroelastic material that uses a neo-Hookean model for the solid phase.

```
<permeability type="perm-Holmes-Mow">
  <perm>0.001</perm>
  <M>1.5</M>
  <alpha>2</alpha>
</permeability>
```

4.3.2.3. 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}
<perm1>	hydraulic permeability k_{1r}
<perm2>	hydraulic permeability k_{2r}
<M>	exponential strain-dependence coefficient M ($M \geq 0$)
<alpha>	power-law exponent α ($\alpha \geq 0$)

This material uses a strain-dependent permeability tensor that accommodates strain-induced anisotropy:

$$\mathbf{k} = \left(k_{0r} \mathbf{I} + \frac{k_{1r}}{J^2} \mathbf{b} + \frac{k_{2r}}{J^4} \mathbf{b}^2 \right) \left(\frac{J - \varphi_r^s}{1 - \varphi_r^s} \right)^{M(J^2 - 1)/2},$$

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 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>
```

4.3.2.4. 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}
<perm1>	hydraulic permeabilities k_{1r}^a along orthogonal directions ($a = 1, 2, 3$)
<perm2>	hydraulic permeabilities k_{2r}^a along orthogonal directions ($a = 1, 2, 3$)
<M0>	isotropic exponential strain-dependence coefficient M_0 ($M_0 \geq 0$)
<M>	orthotropic exponential strain-dependence coefficient M_a ($a = 1, 2, 3$, $M_a \geq 0$)
<alpha0>	isotropic power-law exponent α_0 ($\alpha_0 \geq 0$)
<alpha>	power-law exponent α_a ($a = 1, 2, 3$, $\alpha_a \geq 0$)

This material uses a strain-dependent permeability tensor that accommodates strain-induced anisotropy:

$$\mathbf{k} = k_0 \mathbf{I} + \sum_{a=1}^3 k_1^a \mathbf{m}_a + k_2^a (\mathbf{m}_a \cdot \mathbf{b} + \mathbf{b} \cdot \mathbf{m}_a),$$

where,

$$k_0 = k_{0r} \left(\frac{J - \varphi_r^s}{1 - \varphi_r^s} \right)^{\alpha_0} e^{M_0(J^2-1)/2}$$

$$k_1^a = \frac{k_{1r}^a}{J^2} \left(\frac{J - \varphi_r^s}{1 - \varphi_r^s} \right)^{\alpha_a} e^{M_a(J^2-1)/2}, \quad a = 1, 2, 3,$$

$$k_2^a = \frac{k_{2r}^a}{2J^4} \left(\frac{J - \varphi_r^s}{1 - \varphi_r^s} \right)^{\alpha_a} e^{M_a(J^2-1)/2}, \quad a = 1, 2, 3$$

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). Note that the permeability in the reference state ($\mathbf{F} = \mathbf{I}$) is given by

$$\mathbf{k} = k_{0r} \mathbf{I} + \sum_{a=1}^3 (k_{1r}^a + k_{2r}^a) \mathbf{V}_a \otimes \mathbf{V}_a.$$

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>  
</material>
```

4.3.2.5. 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}
<perm1A>	axial hydraulic permeability k_{1r}^A
<perm2A>	axial hydraulic permeability k_{2r}^A
<perm1T>	transverse hydraulic permeability k_{1r}^T
<perm2T>	transverse hydraulic permeability k_{2r}^T
<M0>	isotropic exponential strain-dependence coefficient M_0 ($M_0 \geq 0$)
<MA>	axial exponential strain-dependence coefficient M_A ($M_A \geq 0$)
<MT>	transverse exponential strain-dependence coefficient M_T ($M_T \geq 0$)
<alpha0>	isotropic power-law exponent α_0 ($\alpha_0 \geq 0$)
<alphaA>	axial power-law exponent α_A ($\alpha_A \geq 0$)
<alphaT>	transverse power-law exponent α_T ($\alpha_T \geq 0$)

This material uses a strain-dependent permeability tensor that accommodates strain-induced anisotropy:

$$\begin{aligned} \mathbf{k} = & k_{0r} \left(\frac{J - \varphi_r^s}{1 - \varphi_r^s} \right)^{\alpha_0} e^{M_0(J^2-1)/2} \mathbf{I} \\ & + \left(\frac{k_{1r}^T}{J^2} (\mathbf{b} - \mathbf{m}) + \frac{k_{2r}^T}{2J^4} [2\mathbf{b}^2 - (\mathbf{m} \cdot \mathbf{b} + \mathbf{b} \cdot \mathbf{m})] \right) \left(\frac{J - \varphi_r^s}{1 - \varphi_r^s} \right)^{\alpha_T} e^{M_T(J^2-1)/2}, \\ & + \left(\frac{1}{J^2} k_{1r}^A \mathbf{m} + \frac{1}{2J^4} k_{2r}^A (\mathbf{m} \cdot \mathbf{b} + \mathbf{b} \cdot \mathbf{m}) \right) \left(\frac{J - \varphi_r^s}{1 - \varphi_r^s} \right)^{\alpha_A} e^{M_A(J^2-1)/2} \end{aligned}$$

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

$$\mathbf{m} = \mathbf{F} \cdot (\mathbf{V} \otimes \mathbf{V}) \cdot \mathbf{F}^T,$$

where \mathbf{V} is a unit vector along the axial direction (defined as described in Section 4.1.1). Note that the permeability in the reference state ($\mathbf{F} = \mathbf{I}$) is given by

$$\mathbf{k} = (k_{0r} + k_{1r}^T + k_{2r}^T) \mathbf{I} + (k_{1r}^A - k_{1r}^T + k_{2r}^A - k_{2r}^T) (\mathbf{V} \otimes \mathbf{V}).$$

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>
```

4.4. 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 compressible materials provided in Section 4.1.3. 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 ($0 < \kappa \leq 1$). 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

$$\mu = \mu_0(\theta) + \frac{R\theta}{M} \ln \frac{c}{\tilde{\kappa}}.$$

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

$$\tilde{\mu}^w = \mu_0^w(\theta) + \frac{1}{\rho_r^w}(p - R\theta\Phi c),$$

where μ_0^w is the solvent chemical potential at some reference temperature θ ; ρ_r^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 dependence 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

$$\begin{aligned}\tilde{p} &= p - R\theta\Phi c \\ \tilde{c} &= \frac{c}{\tilde{\kappa}}.\end{aligned}$$

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

$$\begin{aligned}\mathbf{w} &= -\tilde{\mathbf{k}} \cdot \left(\nabla \tilde{p} + R\theta \frac{\tilde{\kappa}}{d_0} \mathbf{d} \cdot \nabla \tilde{c} \right) \\ \mathbf{j} &= \tilde{\kappa} \mathbf{d} \cdot \left(-\varphi^w \nabla \tilde{c} + \frac{\tilde{c}}{d_0} \mathbf{w} \right),\end{aligned}$$

where

$$\tilde{\mathbf{k}} = \left[\mathbf{k}^{-1} + \frac{R\theta}{\varphi^w} \frac{\tilde{\kappa}c}{d_0} \left(\mathbf{I} - \frac{\mathbf{d}}{d_0} \right) \right]^{-1}$$

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.

4.4.1. Guidelines for Biphasic-Solute Analyses

4.4.1.1. 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 3.8.3) 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.

4.4.1.2. 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 4.5.1.2.

4.4.2. 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 Φ
<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 4.1.3. 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 4.3.2.

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 3.10.2). 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 4.4.3 and 4.4.4, 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 and absolute temperature θ 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.

4.4.3. 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.

4.4.3.1. Constant Isotropic Diffusivity

The material type for constant isotropic diffusivity materials is “*diff-const-iso*”. The following material parameters must be defined:

<code><free_diff></code>	free diffusivity d_0 of solute (diffusivity in solution)
<code><diff></code>	constant isotropic diffusivity d of solute in biphasic-solute mixture

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 $d \leq d_0$, 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>
```

4.4.3.2. 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)
<diff>	diffusivities d^a along orthogonal directions ($a = 1, 2, 3$)

When the permeability is orthotropic,

$$\mathbf{d} = \sum_{a=1}^3 d^a \mathbf{V}_a \otimes \mathbf{V}_a$$

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 $d^a \leq d_0$, 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>
```

4.4.3.3. 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
<diff0>	diffusivity d_{0r}
<diff1>	diffusivity d_{1r}
<diff2>	diffusivity d_{2r}
<M>	exponential strain-dependence coefficient M ($M \geq 0$)
<alpha>	power-law exponent α ($\alpha \geq 0$)

This material uses a strain-dependent diffusivity tensor that accommodates strain-induced anisotropy:

$$\mathbf{d} = \left(d_{0r} \mathbf{I} + \frac{d_{1r}}{J^2} \mathbf{b} + \frac{d_{2r}}{J^4} \mathbf{b}^2 \right) \left(\frac{J - \varphi_r^s}{1 - \varphi_r^s} \right)^{M(J^2-1)/2},$$

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>
```

4.4.3.4. 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
<diff0>	isotropic diffusivity d_{0r}
<diff1>	diffusivities d_{1r}^a along orthogonal directions ($a = 1, 2, 3$)
<diff2>	diffusivities d_{2r}^a along orthogonal directions ($a = 1, 2, 3$)
<M0>	isotropic exponential strain-dependence coefficient M_0 ($M_0 \geq 0$)
<M>	orthotropic exponential strain-dependence coefficient M_a ($a = 1, 2, 3, M_a \geq 0$)
<alpha0>	isotropic power-law exponent α_0 ($\alpha_0 \geq 0$)
<alpha>	power-law exponent α_a ($a = 1, 2, 3, \alpha_a \geq 0$)

This material uses a strain-dependent diffusivity tensor that accommodates strain-induced anisotropy:

$$\mathbf{d} = d_0 \mathbf{I} + \sum_{a=1}^3 d_1^a \mathbf{m}_a + d_2^a (\mathbf{m}_a \cdot \mathbf{b} + \mathbf{b} \cdot \mathbf{m}_a),$$

where,

$$d_0 = d_{0r} \left(\frac{J - \varphi_r^s}{1 - \varphi_r^s} \right)^{\alpha_0} e^{M_0(J^2-1)/2}$$

$$d_1^a = \frac{d_{1r}^a}{J^2} \left(\frac{J - \varphi_r^s}{1 - \varphi_r^s} \right)^{\alpha_a} e^{M_a(J^2-1)/2}, \quad a = 1, 2, 3,$$

$$d_2^a = \frac{d_{2r}^a}{2J^4} \left(\frac{J - \varphi_r^s}{1 - \varphi_r^s} \right)^{\alpha_a} e^{M_a(J^2-1)/2}, \quad a = 1, 2, 3$$

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). Note that the permeability in the reference state ($\mathbf{F} = \mathbf{I}$) is given by

$$\mathbf{k} = k_{0r} \mathbf{I} + \sum_{a=1}^3 (k_{1r}^a + k_{2r}^a) \mathbf{V}_a \otimes \mathbf{V}_a.$$

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>
```

4.4.4. Solubility Materials

4.4.4.1. 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>
```

4.4.5. Osmotic Coefficient Materials

4.4.5.1. 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>
```

4.5. Triphasic 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. 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 triphasic 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 compressible materials provided in Section 4.1.3. 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 the triphasic theory.

Due to steric volume exclusion and short-range electrostatic interactions, a solute α (where $\alpha = +$ for the cation and $\alpha = -$ for the anion) 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 ($0 < \kappa^\alpha \leq 1$). 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

$$\mu^\alpha = \mu_0^\alpha(\theta) + \frac{R\theta}{M^\alpha} \ln \frac{c^\alpha}{\hat{\kappa}^\alpha}.$$

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 triphasic material is charged and its charge density is 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,

$$c^F = \frac{1 - \varphi_r^s}{J - \varphi_r^s} c_r^F,$$

where φ_r^s is the solid volume fraction and c_r^F is the fixed charge density in the reference configuration.

In the triphasic 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,

$$c^F + z^+ c^+ + z^- c^- = 0,$$

where $z^+ = +1$ is the charge number of the cation and $z^- = -1$ is the charge number of the anion. 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

$$\tilde{\mu}^\alpha = \mu_0^\alpha(\theta) + \frac{R\theta}{M^\alpha} \ln \frac{c^\alpha}{\hat{\kappa}^\alpha} + \frac{z^\alpha}{M^\alpha} F_c \psi.$$

In this expression, F_c represents Faraday's constant. It is also possible to rearrange this expression as

$$\tilde{\mu}^\alpha = \mu_0^\alpha(\theta) + \frac{R\theta}{M^\alpha} \ln \left[\exp \left(z^\alpha \frac{F_c \psi}{R\theta} \right) \frac{c^\alpha}{\hat{\kappa}^\alpha} \right].$$

In a triphasic 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

$$\tilde{\mu}^w = \mu_0^w(\theta) + \frac{1}{\rho_T^w} \left[p - R\theta \Phi(c^+ + c^-) \right],$$

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 triphasic 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 triphasic mixture. In general, momentum and energy balances evaluated across a boundary surface in a triphasic 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}^α as

$$\begin{aligned}\tilde{p} &= p - R\theta\Phi(c^+ + c^-), \\ \tilde{c}^\alpha &= c^\alpha / \tilde{\kappa}^\alpha\end{aligned},$$

where

$$\tilde{\kappa}^\alpha = \hat{\kappa}^\alpha \exp\left(-z^\alpha \frac{F_c \psi}{R\theta}\right)$$

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}^α . 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 triphasic 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}^α is the molar flux of solute α relative to the solid. In general, \mathbf{w} and \mathbf{j}^α are given by

$$\begin{aligned}\mathbf{w} &= -\tilde{\mathbf{k}} \cdot \left(\nabla \tilde{p} + R\theta \sum_{\alpha=+,-} \frac{\tilde{\kappa}^\alpha}{d_0^\alpha} \mathbf{d}^\alpha \cdot \nabla \tilde{c}^\alpha \right), \\ \mathbf{j}^\alpha &= \tilde{\kappa}^\alpha \mathbf{d}^\alpha \cdot \left(-\phi^w \nabla \tilde{c}^\alpha + \frac{\tilde{c}^\alpha}{d_0^\alpha} \mathbf{w} \right), \quad \alpha = +, -\end{aligned}$$

where

$$\tilde{\mathbf{k}} = \left[\mathbf{k}^{-1} + \frac{R\theta}{\phi^w} \sum_{\alpha=+,-} \frac{\tilde{\kappa}^\alpha c^\alpha}{d_0^\alpha} \left(\mathbf{I} - \frac{\mathbf{d}^\alpha}{d_0^\alpha} \right) \right]^{-1}$$

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}^α 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.

4.5.1. Guidelines for Triphasic Analyses

4.5.1.1. Initial State of Swelling

Under traction-free conditions, a triphasic 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 3.7.1), so that they become superposed over and above the initial swelling state.

Example:

```
<Step>
  <Module type="triphasic"/>
  <Control>
    <analysis type="steady-state"/>
    ...
  </Control>
</Step>
<Step>
  <Module type="triphasic"/>
  <Control>
    ...
  </Control>
  <Boundary>
    <prescribe type="relative">
      <node id="22" bc="z" lc="4">1</node>
```

```

    ...
    </prescribe>
  </Boundary>
</Step>

```

4.5.1.2. 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. Thus, based on the electroneutrality condition, the cation and anion concentrations in the external environment are equal to each other, e.g., $c_*^+ = c_*^- \equiv c_*$. It follows that the effective fluid pressure in the external environment is $\tilde{p}_* = -2R\theta\Phi_*c_*$ and the effective concentrations are $\tilde{c}_*^+ = c_*/\hat{\kappa}_*^+$ and $\tilde{c}_*^- = c_*/\hat{\kappa}_*^-$. Therefore, in triphasic analyses, whenever the external environment contains salt ions at a concentration of c_* , 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 3.8.3) 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, $\tilde{c}_*^+ = c_*/\tilde{\kappa}_*^+$ and $\tilde{c}_*^- = c_*/\tilde{\kappa}_*^-$.

4.5.1.3. Prescribed Initial Conditions

When a triphasic material is initially exposed to a given external environment with effective pressure \tilde{p}_* and effective concentrations \tilde{c}_*^α ($\alpha = +, -$), 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}_*^α should be evaluated as described in Section 4.5.1.2.

4.5.1.4. 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 $j_n^- = -I_n/F_c$.

4.5.2. General Specification of Triphasic Materials

The material type for a triphasic material is “*triphasic*”. Constitutive relations must be provided for the solid matrix, the mixture fixed charge density, the hydraulic permeability \mathbf{k} , \mathbf{d} d_0 $\tilde{\kappa}$ the osmotic coefficient Φ , and the properties of each solute. 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
<permeability>	specification of the hydraulic permeability \mathbf{k}
<osmotic_coefficient>	specification of the osmotic coefficient Φ
<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 4.1.3. 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 4.3.2 and 4.4.5.

The <solute> tag provides a description of each solute in the triphasic mixture. It should appear exactly twice in the description of a triphasic material. Each tag includes the required *sol* attribute, which should reference a solute *id* from the <Solutes> description in the <Globals> section (Section 3.10.2). The following parameters must be defined in this description:

<diffusivity>	specification of the solute diffusivities \mathbf{d}^α 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 4.4.3 and 4.4.4, respectively. Each solute tag must include a “sol” attribute

Example:

```
<material id="2" name="Media" type="triphasic">
  <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 triphasic material is employed in an analysis, it is also necessary to specify the values of the universal gas constant R , absolute temperature θ , and Faraday's constant F_c in the <Globals> section, using a self-consistent set of units.

Example:

```

<Globals>
  <Constants>
    <R>8.314e-6</R>
    <T>298</T>
    <Fc>96845e-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 triphasic 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 triphasic material descriptions. Generally, two adjoining triphasic regions may share the same two solutes (e.g., Na and Cl in both regions), in which case these solutes may transport freely across the interface separating these regions; they may share only one solute (e.g, Na and Cl in one region, K and Cl in the other region), in which case only the shared solute may transport across the interface; or they may share no solute, in which case the interface is impermeable to all solutes.

4.6. 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
<center_of_mass>	Position of the center of mass
<E>	Young’s modulus
<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 ν 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^{**}. 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 need the *Constraints* section. See section 3.9.1 for more information.

Example:

```
<material id="1" type="rigid body">
  <density>1.0</density>
  <center_of_mass>0,0,0</center_of_mass>
</material>
```

^{**} This is different from previous versions of FEBio where rigid bodies were initially fully constrained.

Chapter 5 Restart Input file

As of version 1.1.6, FEBio will output a restart file. Two files are created when the restart flag is activated: the binary dump file and the text restart input file.

The restart feature in FEBio allows the user to continue a previously terminated analysis. In addition, it allows the user to modify some of the parameters during the restart. To activate the restart feature, define the *restart* element in the Control section:

```
<Control>
  <restart [file="<dump file>"]>1</restart>
  ...
</Control>
```

This describes the format of the restart input file. This file is used to redefine some parameters when restarting a previously terminated run. 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="1.0">
```

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 some loadcurves.

All sections are optional except for the Archive section, and need only be defined when redefining parameters. In the following paragraphs we describe the different sections in more detail.

5.1. 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>
```

5.2. 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

5.3. The LoadData Section

In the LoadData section the user can redefine some or all of the load curves. The syntax is identical to the LoadData section of the FEBio input file:

```
<LoadData>
  <loadcurve id="n">
    <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.

5.4. Example

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>
```

Chapter 6 Multi-step Analysis

As of version 1.1.6, multi-step problems can be solved with FEBio. A multi-step analysis is defined using multiple steps, where in each step the user can redefine control parameters and boundary conditions. This is useful, for instance, for defining time-dependant boundary conditions or for switching between different analysis types during the simulation.

6.1. The Step section

The multi-step analysis feature introduces a new section to the input file. Each step requires its own *step* section, preferably at the bottom of the input file. In this step section, the user can redefine the control section and the boundary section. The following format is suggested when defining a multi-step analysis:

```
<febio_spec version="1.0">
  <Control>
    <!-- global control parameters -->
  </Control>
  <Material>
    <!-- materials go here -->
  </Material>
  <Geometry>
    <!-- geometry goes here -->
  </Geometry>
  <Boundary>
    <!-- global boundary conditions -->
  </Boundary>
  <LoadData>
    <!-- load curve data goes here -->
  </LoadData>
  <Step>
    <Control>
      <!-- local control settings -->
    </Control>
    <Boundary>
      <!-- local boundary conditions -->
    </Boundary>
  </Step>
</febio_spec>
```

The first part of the file looks similar to a normal input file, except that in the control section only global control parameters should be defined (e.g. the title). Also, the boundary section should only contain global boundary conditions. The differences between local and global parameters are explained below.

After the LoadData section, the user can define as many Step sections as needed. In each Step section, the user can now define the (local) control parameters and boundary conditions.

In a multi-step analysis it is important to understand the difference between local and global settings. The global settings are those settings that remain unchanged during the entire simulation.

6.1.1. Control Settings

All control settings are considered local, except the title. Therefore the title should be the only control setting that is defined in the global control section. All other control parameters, including time step parameters, linear solver parameters, convergence parameters, and so forth, should be defined in the Control section of each step. For example:

```
<febio_spec version="1.0">
  <Control>
    <title>This is the title</title>
  </Control>
  ...
  <Step>
    <Control>
      <!-- place control parameters here -->
    </Control>
  </Step>
</febio_spec>
```

6.1.2. Boundary and Loads Settings

The Boundary and Loads sections define the boundary conditions that are applied to the model. If it is defined before the first *Step* section, the conditions remain enforced during the entire simulation. If a Boundary and/or Loads section is defined in a Step section, then the boundary conditions will remain active only during this step. Currently, the only types of boundary conditions that can be enforced this way are the prescribed displacement, nodal forces and rigid contact boundary conditions.

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 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="1.0">
  <Control>
    <title>Multi-step example</title>
  </Control>
  <Material>
```

```

    <material id="1" name="Material 1" type="neo-Hookean">
      <density>1.0</density>
      <E>1</E>
      <v>0.45</v>
    </material>
  </Material>
  <Geometry>
    <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>
      <hex8 id="1" mat="1">1,5,7,3,2,6,8,4</hex8>
    </Elements>
  </Geometry>
  <Boundary>
    <fix>
      <node id="1" bc="xyz"></node>
      <node id="2" bc="xy"></node>
      <node id="3" bc="xz"></node>
      <node id="4" bc="x"></node>
      <node id="5" bc="yz"></node>
      <node id="6" bc="y"></node>
      <node id="7" bc="z"></node>
    </fix>
  </Boundary>
  <LoadData>
    <loadcurve id="1">
      <loadpoint>0,0</loadpoint>
      <loadpoint>1,0.1</loadpoint>
    </loadcurve>
  </LoadData>
  <Step>
    <Control>
      <time_steps>10</time_steps>
      <step_size>0.1</step_size>
    </Control>
    <Boundary>
      <prescribe>
        <node id="5" bc="x" lc="1">1</node>
        <node id="6" bc="x" lc="1">1</node>
        <node id="7" bc="x" lc="1">1</node>
        <node id="8" bc="x" lc="1">1</node>
      </prescribe>
    </Boundary>
  </Step>

```

```
<Step>
  <Control>
    <time_steps>50</time_steps>
    <step_size>0.5</step_size>
    <analysis type="dynamic"></analysis>
  </Control>
</Step>
</febio_spec>
```

Chapter 7 Parameter Optimization

This chapter describes FEBio’s parameter optimization module. This module tries to estimate material parameters by solving an inverse finite element problem, where the “solution” of the problem is known, and the problem’s material parameters are sought. In this case, the solution will often 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. The user will need to input this reaction force curve as well as an initial guess for the material parameters. This is done using a separate input file which 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:

- *Model*: defines the FE problem input file.
- *Options*: optional section defining the optimization control parameters
- *Function*: defines the parameter that generates the function values of the solution that are to be fitted.
- *Parameters*: defines the material parameters that are to be determined
- *LoadData*: defines the experimental reaction force using load curves.

In the following paragraphs each section will be explained in detail.

7.1.1. Model section

The model section defines the file name for the FE input problem. This input file defines the problem that will be solved by FEBio repeatedly. This xml-tag only takes a single value, namely the name of a valid FEBio input file. For example,

```
<Model>ex1.feb</Model>
```

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. Since only one optimization method is currently available, the attribute can be omitted. Note that this section is optional. When omitted, default values for the control parameters are chosen. 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

Comments:

1. The objective function that is to be minimized is a χ^2 function of the reaction force values:

$$\chi^2(\mathbf{a}) = \sum_{i=1}^n \left[\frac{y_i - y(x_i; \mathbf{a})}{\sigma_i} \right]^2.$$

Here, $y(x; \mathbf{a})$ is the function that describes the model, \mathbf{a} is a vector with the (unknown) material parameters and the (x_i, y_i) are the experimentally (or otherwise) obtained data with standard deviation σ_i that approximates the ideal model.

2. The optimization method currently implemented requires the calculation of the gradient of the model function y with respect to the model parameters \mathbf{a} . Since this gradient is not known, FEBio will approximate it using forward differences. For example, the k -th component of the gradient is approximated as follows.

$$\frac{\partial y}{\partial a_k} \approx \frac{1}{\delta a_k} \left[y(a_1, \dots, a_k + \delta a_k, \dots, a_m) - y(a_1, \dots, a_k, \dots, a_m) \right]$$

The value for δa_k is determined from the following formula.

$$\delta a_k = \varepsilon(1 + a_k)$$

where, ε is the forward difference scale factor which can be set by the user with the `fdiff_scale` option.

7.1.3. Function section

This section defines the parameter that will generate the function values that are to be fitted to the experimental data. Currently, only one function parameter can be defined and it is defined using the `fnc` tag. For example,

```
<Function>
  <fnc lc="1">rigid.Fx</fnc>
</Function>
```

This example defines the x -component of a rigid body material named *rigid* as the model's function parameter. The `lc` attribute defines the loadcurve number that describes the model's experimental data.

Note that currently only rigid body materials can be used to define the function parameter, and only using the following data components.

Component	Description
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 moment
My	y-component of the rigid body reaction moment
Mz	z-component of the rigid body reaction moment

7.1.4. 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 format of the name depends on the nesting of the material description in the input file. FEBio supports names in the following form:

```
material_name.parameter_name (standard material)
material_name.elastic.parameter_name (nested material)
material_name.solid_name.parameter_name (solid mixture)
material_name.solid.parameter_name (biphasic material)
material_name.permeability.parameter_name (biphasic material)
material_name.solid.solid_name.parameter_name (biphasic material
with solid mixture)
```

Furthermore, if a parameter is part of an array, its name should be specified in the form *parameter_name*[*index*], where the index is one-based.

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. 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, for a neo-Hookean solid,

```
<Parameters>
  <param name="mat1.E">1.0, 0.0, 5.0, 1.0</param>
  <param name="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 (defined in the *Model* section). The second component (here *E* and ν) are the names of the material parameters.

For a solid mixture that includes an ellipsoidal fiber distribution,

```
<Parameters>
  <param name="mat1.mat2.ksi[1]">0.2, 0, 6, 1.0</param>
</Parameters>
```

In this example, the name of the solid mixture is *mat1*; the name of the ellipsoidal fiber distribution solid is *mat2*; and *ksi[1]* is the first material parameter in an array of three values for this parameter. In order for this naming convention to work correctly, the user must ensure that a unique name is given for each solid in a solid mixture.

For a biphasic material, the parameter name for material properties of the `<solid>` and `<permeability>` materials can be referenced as in this example,

```
<Parameters>
  <param name="mat1.phi0">0.2, 0.01, 0.99, 0.5</param>
  <param name="mat1.solid.E">0.1, 0.01, 0.5, 1.0</param>
  <param name="mat1.permeability.perm">2e-3,5e-4,1e-2,1e-3</param>
</Parameters>
```

Note that *phi0* is a material parameter that belongs to the biphasic material description, whereas *E* belongs to the solid description and *perm* belongs to the permeability description.

7.1.5. LoadData Section

This section serves the same purpose as in the regular FEBio input file: it defines all the load curves that are used in the parameter optimization. Currently, only one load curve needs to be defined, namely the experimental data to which the model will be fitted. The format is identical to that of the usual FEBio input file (see section 3.11).

Each load curve is defined through an array of value pairs (x_i, y_i) , where the *x* component refers to the (simulation) time and the *y* component to is the function value.

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 geometry, materials, boundary conditions, etc. The second input file describes the parameter optimization data, such as the objective and which material 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 -s optim.feb
```

Comments:

1. To run an optimization file, you need the `-s` command line option instead of the `-i` option which runs a regular FEBio input file.
2. The input file (here *optim.feb*) is the name of the parameter optimization input file. The FE model input file does not need to be specified here, since it is specified in the optimization input file.

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>
  <Model>ex1.feb</Model>
  <Options>
    <obj_tol>0.001</obj_tol>
    <f_diff_scale>0.001</f_diff_scale>
  </Options>
  <Function>
    <fnc lc="1">rigid.Fx</fnc>
  </Function>
  <Parameters>
    <param name="mat1.E">1, 0, 5</param>
    <param name="mat1.v">-0.5, 0, 0.5</param>
  </Parameters>
  <LoadData>
    <loadcurve id="1">
      <point>0.0, 0</point>
      <point>0.5, 1</point>
      <point>1.0, 2</point>
    </loadcurve>
  </LoadData>
</febio_optimize>
```

Comments:

1. Notice that the xml root element is *febio_optimize* for the optimization input file.
2. Here, the FEBio input file that contains the actual FE model data is *ex1.feb*. This file is a standard FEBio input file that defines all geometry, materials, boundary conditions and more.
3. The *Options* section is included here, but can be omitted. If omitted default values will be used for all control parameters.
4. The function parameter and material parameters are defined through “material name”.”parameter name” pairs. The name of the materials and their parameters are defined in the FEBio input file (in this example, *ex1.feb*).

Appendix A. Configuration file

As of version 1.2, 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 “1.0”. The following elements are defined.

Parameter	Description
linear_solver	Set the default linear solver for the platform. (1)

Comments:

1. FEBio supports several linear solvers, such as SuperLU, 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 SuperLU and the Pardiso solver are available on most platforms.

Appendix B. LSDYNA Database Format

As of version 1.4 the LSDYNA format is no longer the preferred output format for FEBio plotfiles. For backward compatibility, users can still use this format. To use it, define the following element in the *Output* section of the FEBio input file.

```
<plotfile type="lsdyna"/>
```

This format offers predetermined data storage options. However, in order to provide the user with additional flexibility, FEBio allows the user to customize the data fields in the plotfile. By default, FEBio will choose the data fields, but you can override them by adding the *plotfile* tag in the Output section of your input file.

Currently, FEBio allows you to redefine the following predefined data fields.

<i>LSDYNA data field</i>	<i>FEBio data field</i>
displacement	DISPLACEMENT
velocity	NONE
	VELOCITY
	FLUID_FLUX
	CONTACT_TRACTION
	REACTION_FORCE
	MATERIAL_FIBER
acceleration	NONE
	ACCELERATION
	FLUID_FLUX
	CONTACT_TRACTION
	REACTION_FORCE
	MATERIAL_FIBER
temperature	NONE
	FLUID_PRESSURE
	CONTACT_PRESSURE
	CONTACT_GAP
plastic strain	PLASTIC_STRAIN
	FIBER_STRAIN
	DEV_FIBER_STRAIN

The following example illustrates how to map FEBio data fields to LSDYNA data fields:

```
<Output>
  <plotfile>
    <map field="displacement">DISPLACEMENT</map>
    <map field="velocity">CONTACT_TRACTION</map>
    <map field="acceleration">REACTION_FORCE</map>
    <map field="temperature">CONTACT_GAP</map>
  </plotfile>
</Output>
```

References

- [1] Maker, B. N., 1995, "NIKE3D: A nonlinear, implicit, three-dimensional finite element code for solid and structural mechanics," Lawrence Livermore Lab Tech Rept, UCRL-MA-105268.
- [2] Gee, M. W., Dohrmann, C. R., Key, S. W., and Wall, W. A., 2009, "A uniform nodal strain tetrahedron with isochoric stabilization," *Int. J. Numer. Meth. Engng*(78), pp. 429-443.
- [3] Laursen, T. A., and Maker, B. N., 1995, "Augmented Lagrangian quasi-newton solver for constrained nonlinear finite element applications," *International Journal for Numerical Methods in Engineering*, 38(21), pp. 3571-3590.
- [4] Ateshian, G., Maas, S., and Weiss, J. A., 2009, "Finite Element Algorithm for Frictionless Contact of Porous Permeable Media under Finite Deformation and Sliding," *J. Biomech. Engn.*
- [5] Simo, J. C., and Taylor, R. L., 1991, "Quasi-incompressible finite elasticity in principal stretches: Continuum basis and numerical algorithms," *Computer Methods in Applied Mechanics and Engineering*, 85, pp. 273-310.
- [6] Arruda, E. M., and Boyce, M. C., 1993, "A Three-Dimensional Constitutive Model for the Large Stretch Behavior of Rubber Elastic Materials," *J. Mech. Phys. Solids*, 41(2), pp. 389-412.
- [7] Lanir, Y., 1983, "Constitutive equations for fibrous connective tissues," *J Biomech*, 16(1), pp. 1-12.
- [8] Ateshian, G. A., Rajan, V., Chahine, N. O., Canal, C. E., and Hung, C. T., 2009, "Modeling the matrix of articular cartilage using a continuous fiber angular distribution predicts many observed phenomena," *J Biomech Eng*, 131(6), p. 061003.
- [9] Ateshian, G. A., 2007, "Anisotropy of fibrous tissues in relation to the distribution of tensed and buckled fibers," *J Biomech Eng*, 129(2), pp. 240-249.
- [10] Fung, Y. C., 1993, *Biomechanics : mechanical properties of living tissues*, Springer-Verlag, New York.
- [11] Fung, Y. C., Fronek, K., and Patitucci, P., 1979, "Pseudoelasticity of arteries and the choice of its mathematical expression," *Am J Physiol*, 237(5), pp. H620-631.
- [12] Ateshian, G. A., and Costa, K. D., 2009, "A frame-invariant formulation of Fung elasticity," *J Biomech*, 42(6), pp. 781-785.
- [13] Blemker, S., 2004, "3D Modeling of Complex Muscle Architecture and Geometry," Stanford University, Stanford.
- [14] Criscione, J., Douglas, S., and Hunter, W., 2001, "Physically based strain invariant set for materials exhibiting transversely isotropic behavior," *J. Mech. Phys. Solids*, 49, pp. 871-897.
- [15] Spencer, A. J. M., 1984, *Continuum Theory of the Mechanics of Fibre-Reinforced Composites*, Springer-Verlag, New York.
- [16] Ateshian, G. A., Ellis, B. J., and Weiss, J. A., 2007, "Equivalence between short-time biphasic and incompressible elastic material response," *J Biomech Eng*, In press.
- [17] Puso, M. A., and Weiss, J. A., 1998, "Finite element implementation of anisotropic quasi-linear viscoelasticity using a discrete spectrum approximation," *J Biomech Eng*, 120(1), pp. 62-70.
- [18] Quapp, K. M., and Weiss, J. A., 1998, "Material characterization of human medial collateral ligament," *J Biomech Eng*, 120(6), pp. 757-763.

- [19] Weiss, J. A., Maker, B. N., and Govindjee, S., 1996, "Finite element implementation of incompressible, transversely isotropic hyperelasticity," *Computer Methods in Applications of Mechanics and Engineering*, 135, pp. 107-128.
- [20] Veronda, D. R., and Westmann, R. A., 1970, "Mechanical Characterization of Skin - Finite Deformations," *J. Biomechanics*, Vol. 3, pp. 111-124.
- [21] Overbeek, J. T., 1956, "The Donnan equilibrium," *Progress in biophysics and biophysical chemistry*, 6, pp. 57-84.
- [22] Lai, W. M., Hou, J. S., and Mow, V. C., 1991, "A triphasic theory for the swelling and deformation behaviors of articular cartilage," *Journal of biomechanical engineering*, 113(3), pp. 245-258.
- [23] Holmes, M. H., and Mow, V. C., 1990, "The nonlinear characteristics of soft gels and hydrated connective tissues in ultrafiltration," *J Biomech*, 23(11), pp. 1145-1156.
- [24] Ateshian, G. A., Warden, W. H., Kim, J. J., Grelsamer, R. P., and Mow, V. C., 1997, "Finite deformation biphasic material properties of bovine articular cartilage from confined compression experiments," *J Biomech*, 30(11-12), pp. 1157-1164.
- [25] Iatridis, J. C., Setton, L. A., Foster, R. J., Rawlins, B. A., Weidenbaum, M., and Mow, V. C., 1998, "Degeneration affects the anisotropic and nonlinear behaviors of human annulus fibrosus in compression," *J Biomech*, 31(6), pp. 535-544.
- [26] Bonet, J., and Wood, R. D., 1997, *Nonlinear continuum mechanics for finite element analysis*, Cambridge University Press.
- [27] Mow, V. C., Kuei, S. C., Lai, W. M., and Armstrong, C. G., 1980, "Biphasic creep and stress relaxation of articular cartilage in compression: Theory and experiments," *Journal of biomechanical engineering*, 102(1), pp. 73-84.
- [28] Mow, V. C., Kwan, M. K., Lai, W. M., and Holmes, M. H., 1985, "A finite deformation theory for nonlinearly permeable soft hydrated biological tissues," *Frontiers in Biomechanics*, G. a. W. Schmid-Schonbein, SL-Y and Zweifach, BW, ed., pp. 153-179.