

# **FEBio**

**FINITE ELEMENTS FOR BIOMECHANICS**

*Version 1.1.7*

## **User's Manual**

Written and Edited by

*Steve Maas, Dave Rawlins, Jeff Weiss*

**Musculoskeletal Research Laboratories  
Department of Bioengineering, and  
Scientific Computing and Imaging Institute  
University of Utah**

**72 S. Central Campus Drive, Room 2646  
Salt Lake City, Utah**

[steve.maas@utah.edu](mailto:steve.maas@utah.edu)

[rawlins@sci.utah.edu](mailto:rawlins@sci.utah.edu)

[jeff.weiss@utah.edu](mailto:jeff.weiss@utah.edu)

**Last Updated: June 23, 2009**

# Table of Contents

<b>CHAPTER 1 - INTRODUCTION.....</b>	<b>4</b>
1.1. OVERVIEW OF FEBIO.....	4
1.2. ABOUT THIS DOCUMENT .....	5
<b>CHAPTER 2 - RUNNING FEBIO.....</b>	<b>6</b>
2.1. THE COMMAND LINE .....	6
2.2. THE CONFIGURATION FILE .....	7
2.3. FEBIO OUTPUT .....	7
2.4. ADVANCED OPTIONS .....	7
2.4.1. <i>Interrupting a run</i> .....	7
2.4.2. <i>Debugging a run</i> .....	8
2.4.3. <i>Restarting a run</i> .....	8
2.4.4. <i>Input file checking</i> .....	8
<b>CHAPTER 3 - FREE FORMAT INPUT.....</b>	<b>9</b>
3.1. FREE FORMAT OVERVIEW .....	9
3.2. CONTROL SECTION .....	11
3.2.1. <i>Mandatory Parameters</i> .....	11
3.2.2. <i>Optional Parameters</i> .....	11
3.3. MATERIAL SECTION.....	14
3.3.1. <i>Incompressible materials</i> .....	15
3.3.2. <i>Specifying fiber orientation</i> .....	15
3.3.3. <i>Linear Elastic</i> .....	19
3.3.4. <i>St. Venant-Kirchhoff</i> .....	20
3.3.5. <i>Neo-Hookean</i> .....	21
3.3.6. <i>Mooney-Rivlin</i> .....	22
3.3.7. <i>Ogden</i> .....	23
3.3.8. <i>Arruda-Boyce</i> .....	24
3.3.9. <i>Veronda-Westmann</i> .....	25
3.3.10. <i>Transversely Isotropic Mooney-Rivlin</i> .....	26
3.3.11. <i>Transversely Isotropic Veronda-Westmann</i> .....	28
3.3.12. <i>Poroelasticity</i> .....	29
3.3.13. <i>Rigid Body</i> .....	30
3.3.14. <i>Tension-Compression Nonlinear Orthotropic</i> .....	32
3.3.15. <i>Muscle Material</i> .....	33
3.3.16. <i>Tendon Material</i> .....	35
3.4. GEOMETRY SECTION.....	36
3.4.1. <i>Nodes Section</i> .....	36
3.4.2. <i>Elements Section</i> .....	36
3.4.3. <i>ElementData Section</i> .....	38
3.5. BOUNDARY SECTION .....	39
3.5.1. <i>Prescribed Nodal Degrees of Freedom</i> .....	39
3.5.2. <i>Nodal forces</i> .....	39
3.5.3. <i>Pressure forces</i> .....	40
3.5.4. <i>Contact Interfaces</i> .....	40
3.6. GLOBALS SECTION.....	45
3.7. LOADDATA SECTION .....	46
3.8. OUTPUT SECTION.....	47
3.8.1. <i>Logfile</i> .....	47

3.8.2. Plotfile.....	51
<b>CHAPTER 4 - RESTART INPUT FILE.....</b>	<b>53</b>
4.1. THE ARCHIVE SECTION.....	53
4.2. THE CONTROL SECTION.....	53
4.3. THE LOADDATA SECTION.....	54
4.4. EXAMPLE.....	54
<b>CHAPTER 5 - MULTI-STEP ANALYSIS.....</b>	<b>55</b>
5.1. THE STEP SECTION.....	55
5.1.1. Control Settings.....	56
5.1.2. Boundary Settings.....	56
5.2. AN EXAMPLE.....	56
<b>APPENDIX A. FIXED FORMAT INPUT.....</b>	<b>58</b>
A.1. CONTROL SECTION.....	59
A.1.1. Control Line 1.....	59
A.1.2. Control Line 2.....	60
A.1.3. Control Line 3.....	61
A.1.4. Control Line 4.....	62
A.1.5. Control Line 5.....	63
A.1.6. Control Line 6.....	64
A.1.7. Control Line 7.....	65
A.1.8. Control Line 8.....	66
A.1.9. Control Line 9.....	66
A.1.10. Control Line 10.....	66
A.2. MATERIAL SECTION.....	67
A.2.1. Line 1.....	67
A.2.2. Line 2.....	67
A.2.3. Material Type 1 – Neo-Hookean.....	68
A.2.4. Material Type 15 – Mooney Rivlin Hyperelastic.....	68
A.2.5. Material Type 18 – Transversely Isotropic Hyperelastic.....	68
A.2.6. Material type 20 – Rigid Body.....	70
A.3. NODAL COORDINATES SECTION.....	71
A.4. ELEMENT CONNECTIVITY SECTION.....	72
A.5. RIGID NODE AND FACET SECTION.....	73
A.6. CONTACT SECTION.....	74
A.6.1. Control line.....	74
A.6.2. Auxiliary control line.....	74
A.6.3. Slave facet cards.....	75
A.6.4. Master facet cards.....	75
A.7. LOAD CURVE SECTION.....	76
A.7.1. Line 1.....	76
A.7.2. Line 2, ..., pts.....	76
A.8. CONCENTRATED NODAL FORCE SECTION.....	77
A.9. PRESSURE BOUNDARY SECTION.....	78
A.10. PRESCRIBED DISPLACEMENT SECTION.....	79
A.11. BODY FORCE SECTION.....	80
A.11.1. Line 1.....	80
A.11.2. Line 2.....	80
A.11.3. Line 3.....	80
<b>APPENDIX B. CONFIGURATION FILE.....</b>	<b>81</b>
<b>REFERENCES.....</b>	<b>82</b>



# 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 numerous research areas in biomechanics. This section 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.

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

Several nonlinear constitutive models are available, allowing the user to model the often complicated biological tissue behavior. Several isotropic material models are supported such as Neo-Hookean, Mooney-Rivlin, Ogden, Arruda-Boyce and Veronda-Westmann. All these models have a non-linear stress-strain response. A linear elastic model is also available for small strain scenarios and validation problems. 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.

Biological tissues can interact in very complicated ways. Therefore FEBio supports a wide range of boundary conditions to model these interactions. 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. Finally, the user may specify a body force to model the effects of, for instance, gravity or base acceleration.

FEBio does not have any 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 and Abaqus files can be imported in PreView and can be exported 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 FEBio's input format. This input format is an XML-based format that organizes the data in a convenient hierarchical structure. Chapter 4 discussed the restart capabilities of FEBio. The restart feature allows the user to interrupt a run and continue it at a later time, optionally making some changes to the problem data. Chapter 5 describes the multi-step analysis feature where the user can split up the entire analysis in several steps.

For historical reasons FEBio also supports a fixed input format, which is detailed in Appendix A. This format lists the problem data in a predefined sequential list and is almost identical to the input specification of NIKE3D (Lawrence Livermore National Laboratories). Therefore FEBio can run NIKE3D input files, however with some limitations as discussed in Appendix A.

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

### 2.1. The Command Line

FEBio runs on several different computing platforms including Windows XP, Mac OSX and most Linux flavors. FEBio is started from a shell window (also known as 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 in free or fixed format
- `-r` : restart file name
- `-g` : debug flag (does not require a file name)
- `-p`: plot file name
- `-a`: dump file name
- `-d`: diagnostic input file
- `-o`: log file name
- `-c`: data check only
- `-s`: optimization input file
- `-nosplash`: don't show the welcome screen
- `-cnf`: configuration filename
- `-noconfig`: don't use the configuration file

The `-i` option is used to specify the name of the input file. Note that the format of the input file is determined by the file extension. A “.feb” or “.xml” implies that the input file is formatted in the free format. A “.n” extension or no extension indicates that the input file is formatted in the older fixed format.

*Example:* `> febio -i input.feb`

The `-r` option allows you to restart a previously started analysis. The filename that must follow this option is an FEBio *restart input file*. The restart input file is described in more detail in chapter Chapter 4 - . The `-i` and `-r` options are mutually exclusive; only one of them may appear on the command line.

*Example:* `> febio -r file.feb`

The `-g` option runs FEBio in *debug mode*. See section 2.4.2 for more information on running FEBio in debug mode.

*Example:* `> febio -i input.feb -g`

## 2.2. The configuration file

As of version 1.2 FEBio requires a *configuration file* to run. The purpose of the file is 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 B. 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`

## 2.3. 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. In case the run terminates unexpectedly, this file can be used to restart the analysis from the last converged time step. See chapter Chapter 4 - 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. If the input file was in the free format 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*. If the input file was in the fixed format, the following default names are used: *f3plot* for the plot file, *f3log.txt* for the log file and *f3dmp* for the dump file.

## 2.4. Advanced Options

### 2.4.1. Interrupting a run

The user can pause the run by pressing `ctrl+c`. This will bring up a prompt, and the user can enter a command. The following commands are available.

- *cont*: continues the run
- *conv*: force the current time step to converge
- *debug [on/off]*: toggle debug mode
- *dtmin*: set the minimum time step size

- *fail*: stop the current iteration and retry
- *help*: list the available commands with a short explanation
- *plot*: dump current state to plot database and continue
- *print*: print values of variables
- *quit*: exit the application
- *restart*: toggles restart flag
- *version*: print version information

Note that it may take a while before the command prompt is displayed after FEBio detects the *ctrl+c* interruption.

### 2.4.2. Debugging a run

As stated in section 2.1 you can run FEBio 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. Because of this additional work, the problem might run slower.

### 2.4.3. 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 3 for more details.

To restart an analysis, just use the *-r* command line option. This option requires a filename as 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 you to redefine some parameters when restarting the run. The format of this file is described in chapter 4.

### 2.4.4. 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

In this chapter we describe the free input format used by FEBio. This format is based on the popular XML format that is used as a standard in web based data representation. The free input format follows the standard XML conventions and therefore 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.

Before we start with the format specification it is useful to review the syntax of XML and to define some terminology. 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>
```

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*. 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.
- *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 problem steps, where in each problem boundary and initial conditions can be defined.

## 3.2. 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 solver parameters. These parameters are defined as child elements of the *Control* element. The parameters are grouped into two groups: mandatory parameters and optional parameters.

### 3.2.1. Mandatory Parameters

The following parameters must be specified in every input file.

Parameter	Description
time_steps	Total number of time steps that will be taken. (= <i>ntime</i> )
step_size	The initial time step size. (= <i>dt</i> ) (1)

*Comments:*

1. The total running time of the analysis is determined by  $ntime * dt$ .

### 3.2.2. Optional Parameters

The following parameters are optional. If not specified they are assigned the default values, which are found in the last column.

Parameter	Description	Default
title	Title of problem	(none)
dtol	Convergence tolerance on displacements	0.001
etol	Convergence tolerance on energy	0.01
rtol	Convergence tolerance on residual	1e+10
lstol	Convergence tolerance on line search	0.9
time_stepper	Enable the auto time stepper (1)	(off)
max_refs	Max number of stiffness reformations	15
max_ups	Max number of BFGS stiffness updates	10
optimize_bw	Optimize bandwidth of stiffness matrix (2)	0
restart	Generate restart flag (3)	0
plot_level	Sets the level of state dumps to the plot file (4)	1
cmax	Set the max condition number for the stiffness matrix (5)	1e+5
analysis	Sets the analysis type (6)	static
print_level	Sets the amount of output that is generated on screen (7)	0

*Comments:*

1. If the *time\_stepper* parameter is defined it will enable the auto time-stepper, which will adjust the time step size based on convergence information. 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	
opt_iter	Optimal number of iterations	11

The *dtmin* and *dtmax* values are used to constrain the range of possible time step values. The *opt\_iter* defines the 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. The *max\_retries* parameter determines the maximum number of times a timestep may be retried before FEBio error terminates.

- This option allows you to minimize the bandwidth of the global stiffness matrix. This can drastically decrease the memory requirements and running times when using the default skyline solver. It is highly recommended when using this solver.

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

- This flag can be used to request the generation of a restart dump file. To activate it, specify a non-zero value. You can optionally also specify a filename for the dump file. If omitted a default name will be provided. Note that this will only generate the binary dump file. If you wish to use a restart input file, you need 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

- When the condition number of the stiffness matrix becomes too large, the matrix may become ill-conditioned. FEBio monitors this number and when it exceeds *cmax* it reforms the stiffness matrix.

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

- The *analysis* element sets the analysis type. Currently, FEBio defines two analysis types: (quasi-)static and dynamic. In a quasi-static analysis the inertial response is response and an equilibrium solution is sought. Note that in this analysis it is still possible to simulate time dependant effect such as viscoelasticity. In a dynamic analysis the inertial terms are included.

<b>Value</b>	<b>Description</b>
static	(quasi-) static analysis
dynamic	dynamic analysis.

7. The *print\_level* allows the user to control exactly how much output is written to the screen. The following values are allowed.

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

### 3.3. 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="elastic">
```

Or, if the *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. The following material types are available.

- *linear elastic*
- *linear orthotropic*
- *St-Venant-Kirchhoff*
- *neo-Hookean*
- *Mooney-Rivlin*
- *Arruda-Boyce*
- *Ogden*
- *Veronda-Westmann*
- *Holmes-Mow*
- *trans iso Mooney-Rivlin*
- *trans iso Veronda-Westmann*
- *TC nonlinear orthotropic*
- *muscle material*
- *tendon material*
- *Fung orthotropic*
- *viscoelastic*
- *poroelastic*
- *poroelastic Holmes Mow*
- *rigid body*

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*.

### 3.3.1. Incompressible materials

For materials which are considered to be incompressible, the incompressibility constraint is by default enforced using a penalty method that uses the bulk modulus as the penalty factor. However, the constraint can also be enforced more accurately using an augmented Lagrangian method. In order to use this method you need to define two additional material parameters.

<laugon>	Turn augmented Lagrangian on for this material (1) or off (0)
<atol>	Augmentation tolerance

The augmented Lagrangian method for incompressibility enforcement is available for all materials that are based on a decoupled strain energy density function, namely:

- *Mooney-Rivlin*
- *Veronda-Westmann*
- *trans iso Mooney-Rivlin*
- *trans iso Veronda-Westmann*
- *TC nonlinear orthotropic*
- *muscle material*
- *tendon material*
- *Ogden*
- *Arruda-Boyce*

*Example:*

```
<material id="1" type="Mooney-Rivlin">
  <c1>5</c1>
  <c2>0.4</c2>
  <k>10000</k>
  <laugon>1</laugon>      ← turns augmented Lagrangian on
  <atol>0.05</atol>     ← sets the augmentation tolerance
</material>
```

### 3.3.2. Specifying fiber orientation

Some of the anisotropic materials are modeled as an isotropic matrix in which fibers are embedded. For these materials, the user has to specify several fiber parameters, 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 element's fiber directions manually in the *ElementData* section. See section 3.4.3 for more details on how to do this.

## Transversely Isotropic materials

For transversely isotropic materials this 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.

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)

Note that 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.

```
<fiber type="local">1,2</fiber>
```

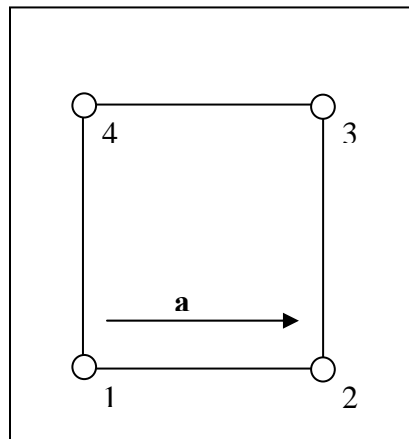


Figure 3-1. *local* fiber direction option .

2. 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>
```

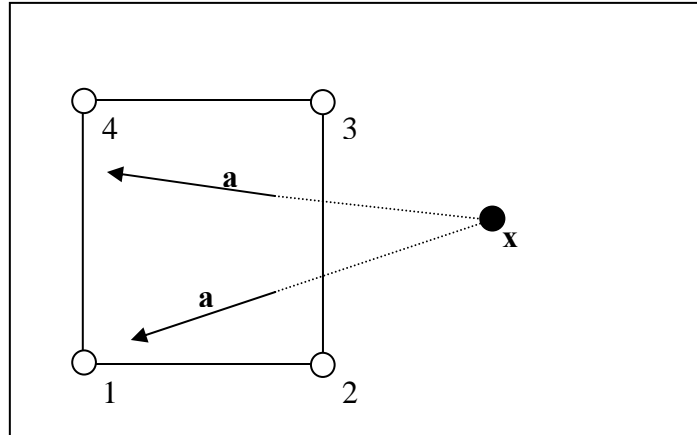


Figure 3-2. *spherical* fiber direction option.

- 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>
```

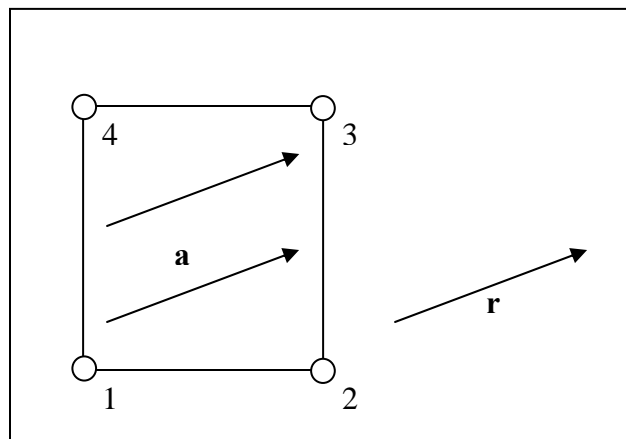


Figure 3-3. *vector* fiber direction option.

- 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 3.4 for more details. In this case the value is ignored.

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

## Orthotropic materials

For orthotropic materials, the user needs to specify two fiber directions  $\mathbf{a}$  and  $\mathbf{d}$ . From these FEBio will generate an orthonormal set of 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 <b>a</b> and <b>d</b> 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>
```

### 3.3.3. Linear Elastic

The material type for isotropic linear elasticity is “*linear elastic*”. The following material parameters must be defined:

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

This is the classical material model used in linear elasticity theory. The Cauchy stress is given by,

$$\sigma_{ij} = \lambda(\text{tr } \boldsymbol{\varepsilon})^2 \delta_{ij} + 2\mu\varepsilon_{ij}.$$

Here,  $\lambda$  and  $\mu$  are the Lamé parameters and are related to the more familiar Young’s modulus  $E$  and Poisson’s ratio  $\nu$  as follows,

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

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

It is often convenient to express the material properties using the bulk modulus  $K$  and shear modulus  $G$ . To convert to Young’s modulus and Poisson’s ratio you can use the following formulas.

$$E = \frac{9KG}{3K + G}$$

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

Note that this material model is only valid for small strains and small rotations. For small strains and large rotations, the St.Venant-Kirchhoff is better suited. It is not recommended using this material for large strains.

*Example:*

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

### 3.3.4. St. Venant-Kirchhoff

The material type for a St. Venant-Kirchhoff material is *St.Venant-Kirchhoff* [1]. The following material parameters must be defined.

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

This is an alternative to linear elasticity. It can only be used for small strains but allows large rotations. Like most of the models used in FEBio it can also be derived from a hyperelastic strain-energy function, namely

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

Here,  $\lambda$  and  $\mu$  are the Lamé parameters and are related to the more familiar Young's modulus  $E$  and Poisson's ratio  $\nu$  as follows,

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

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

The St. Venant-Kirchhoff material is a model of a classical nonlinear material that is often used for metals. Note however, that this material can lead to unrealistic results when used outside the small strain range.

*Example:*

```
<material id="1" type="St.Venant-Kirchhoff">
  <E>1000.0</E>
  <v>0.45</v>
</material>
```

### 3.3.5. 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 [1]. 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 the latter case it is recommended to use the *Mooney-Rivlin* material.

*Example:*

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

### 3.3.6. Mooney-Rivlin

The material type for incompressible 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,

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

$C_1$  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 [2].

This material model is useful for modeling certain types of isotropic biological tissues. For example, the isotropic collagen matrix can be described quite well with this material model.

*Example:*

```
<material id="2" type="Mooney-Rivlin">
  <c1>10.0</c1>
  <c2>20.0</c2>
  <k>1000</k>
</material>
```

### 3.3.7. Ogden

This material describes an incompressible hyperelastic Ogden material. The following material parameters must be defined.

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

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

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

Here,  $\tilde{\lambda}_i^2$  are the eigenvalues of the deviatoric right Cauchy deformation tensor,  $c_i$  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>
```

### 3.3.8. Arruda-Boyce

This material describes an incompressible Arruda-Boyce model [REF]. The following material parameters are required.

mu	Shear modulus
N	Locking stretch
k	Bulk modulus

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

$$W = \mu \sum_{i=1}^5 \frac{C_i}{N^{i-1}} (I_1^i - 3^i) + U(J)$$

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.

This material model was proposed by Arruda and Boyce [REF] and is based on an eight-chain representation of the macromolecular network structure of polymer chains. The strain energy form **Error! Reference source not found.** represent a truncated Taylor series of the inverse Langevin function, which arises in the statistical treatment of non-Gaussian chains.

*Example:*

```
<material id="1" type="Arruda-Boyce">
  <c1>1</c1>
  <c2>1</c2>
  <k>100</k>
</material>
```

### 3.3.9. Veronda-Westmann

The material type for incompressible Veronda-Westmann materials is *Veronda-Westmann*. 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:

$$W = C_1 \left[ e^{(c_2(\tilde{I}_1 - 3))} - 1 \right] - \frac{C_1 C_2}{2} (\tilde{I}_2 - 3) + U(J)$$

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 [3].

*Example:*

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

### 3.3.10. 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 [4-6]. 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 strain energy function can be written as follows:

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

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,  $lam\_max$  ( $\lambda_m$ ) 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  $\lambda_m$ .

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 [2].

The fiber orientation can be specified as explained in section 3.3.2. 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>
```

### 3.3.11. 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 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 3.3.2. 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>
```

### 3.3.12. Poroelasticity

The material type for poroelastic materials is *poroelastic*. The following material parameters must be defined:

<perm>	Permeability
<solid_id>	Material ID of the solid phase

This material model uses the biphasic theory for describing the time-dependent material behavior of materials that consist of both a solid and fluid phase [7, 8]. We refer to the *FEBio Theory Manual* for a more indebt description of the biphasic theory.

This material model uses a “displacement-pressure”, or “u-p” finite element formulation [9]. Solid nodal displacements and fluid nodal pressures are interpolated using trilinear shape functions. Details of this finite element formulation can be found in [10]. At this point the boundary conditions that apply to the pressure degree of freedom are limited to prescribed pressures. See section 3.5 for more details.

A major strength of this implementation of poroelasticity is the seamless integration of solid elements (with degrees of freedom only for the solid displacement) and biphasic elements (which also include degrees of freedom for fluid pressure), thanks to a judicious choice of primary variables and natural boundary conditions in the formulation of the governing equations. Thus, at the interface of a solid element and a biphasic element, continuity of the solid displacement is automatically satisfied, while the fluid flux across this interface automatically reduces to zero, since the solid element is impermeable by definition.

The parameter *perm* defines the permeability of the solid. The *solid\_id* parameter is the material number that is used to describe the solid phase. In other words, to use this material model you need to define at least *two* materials: a material for the solid phase and a poroelastic material. The solid phase may be described by any of the available material models except rigid body and poroelastic.

#### *Example:*

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

```
<material id="1" type="neo-Hookean">
  <E>1.0</E>
  <v>0.4</v>
</material>
<material id="2" type="poroelastic">
  <perm>0.001</perm>
  <solid_id>1</solid_id>
</material>
```

### 3.3.13. 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
<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.

If the *center\_of\_mass* parameter is omitted, FEBio will calculate the center of mass automatically. In this 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 translation and rotation boundary codes are used to specify 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>
```

Type	Description
free	Degree of freedom is moved, i.e. determined by solution
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. Omitting this translation code is the same as specifying a fixed code. The syntax and interpretation is the same for the other translation and rotation codes.

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

*Example:*

```
<material id="5" type="rigid body">
  <center_of_mass>1.0,2.0,3.0</center_of_mass>
  <trans_x type="prescribed" lc="2">2.0</trans_x>
</material>
```

In this example the center of mass is located at (1,2,3) and the x-displacement of the rigid body is prescribed by a load curve. The value of the load curve is scaled by 2. The other translational and rotational degrees of freedom are fixed.

A force (torque) can be applied at 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 can be done as follows.

```
<material id="6" type="rigid body">
  <center_of_mass>0.0,0.0,0.0</center_of_mass>
  <trans_x type="force" lc="1">1.0</trans_x>
  <trans_y type="fixed"></trans_y>
  <trans_z type="fixed"></trans_z>
</material>
```

### 3.3.14. 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 $\beta$ parameter (see below)
<ksi>	the $\xi$ parameter (see below)
<mat_axis>	defines the material axes

This material is based on the following strain energy density function.

$$W(\mathbf{C}, \lambda_1, \lambda_2, \lambda_3) = W_{iso}(\tilde{\mathbf{C}}) + \sum_{i=1}^3 W_i^{TC}(\lambda_i) + U(J)$$

The isotropic strain energy  $W_{iso}$  and the dilatational energy  $U$  are the same as for the Mooney-Rivlin material. The tension-compression term is defined as follows.

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

The  $\lambda_i$  parameters are the fiber stretches of the local material fibers. The local material fibers are defined (in the reference frame) as an orthonormal set of vectors. See section 3.3.2 for more information. 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>
```

### 3.3.15. Muscle Material

This material model implements the constitutive model developed by Silvia S. Blemker [REF]. The model is designed to simulate skeletal muscles. 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 [REF], instead of the usual five Spencer invariants. For this particular material only two of the five Criscione invariants are used. The strain energy function is defined as follows.

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

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  $\alpha$  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>
```

### 3.3.16. 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  $\lambda^*$ .

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="muscle 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>
```

### 3.4. 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

#### 3.4.1. 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 can not be skipped. (So if there is a node 4 and a node 6, there must also be a node 5 somewhere).

#### 3.4.2. Elements Section

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

##### *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.

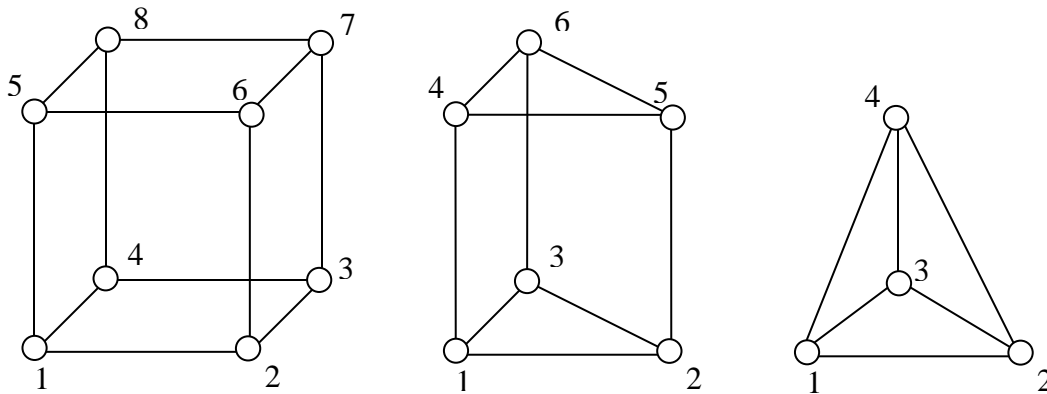


Figure 3-4. Node numbering for solid elements.

### Shell Elements

FEBio currently supports two types of shell elements, namely 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.

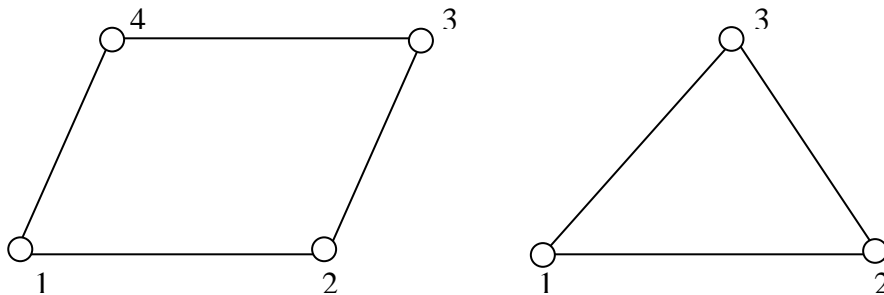


Figure 3-5. Node numbering for shell elements.

### Surface Elements

In many cases the surface (or part of it) of the geometry 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 can not 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;

### 3.4.3. 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 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 3.3 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.

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.

### 3.5. Boundary Section

The *Boundary* section defines all boundary conditions that are applied to the geometry. Several boundary conditions are available: nodal displacements, nodal forces, pressure forces, contact interfaces, rigid joints and linear constraints.

#### 3.5.1. Prescribed Nodal Degrees of Freedom

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

```
<prescribe>
  <node id="n" bc="x" [lc="1"]>1.0</node>
  ...
</prescribe>
```

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 displacement in *x*-direction
- y*: apply displacement in *y*-direction
- z*: apply displacement in *z*-direction
- p*: apply prescribed fluid pressure (only used in poroelastic analysis)

An optional loadcurve can be given as well with the *lc* attribute. If no loadcurve is present the value will be automatically ramped from 0 to the value specified in the xml file.

Degrees of freedom that are fixed (in other words are always zero) can be defined by the *fix* element.

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

In this case the value of the element is ignored. Of course, the *prescribe* element with a value of zero for the *node* tags can also be used to constrain a certain nodal degree of freedom. The advantage of the *fix* element is that the equation corresponding to the constrained degree of freedom is removed from the linear system of equations. This reduces the run time of the FE analysis.

#### 3.5.2. Nodal forces

Nodal forces are applied by the *force* element. These forces always point in the same direction and do not deform with the geometry.

```
<force>
  <node id="n" bc="x" [lc="1"]>1.0</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 fluid flux (only used in poroelastic analysis)

An optional load curve can be given as well with the *lc* attribute. If no loadcurve is present the value will be automatically ramped from 0 to the value specified in the xml file.

### 3.5.3. 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 to the value specified here.

### 3.5.4. Contact Interfaces

Contact boundary conditions are defined with the *contact* element. The *type* attribute specifies what type of contact interface you wish to define.

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

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

Type	Description
sliding_with_gaps	A sliding interface that may separate
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

## Sliding Interfaces

A sliding interface can be used to setup a non-penetration constraint between two surfaces. Each sliding interface consists of a master surface and a slave surface. When using a two-pass algorithm for enforcement of the contact constraint, the definition of master and slave surfaces is arbitrary. 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 master. The following properties are defined for sliding interfaces.

Parameter	Description	Default
tolerance	convergence tolerance	0.01
penalty	normal penalty factor (1)	1
two_pass	two-pass flag (2)	0
laugon	Augmented Lagrangian flag	1
minaug	minimum nr of augmentations	0
maxaug	maximum nr of augmentations	10
fric_coeff	frictional coefficient (3)	0
fric_penalty	tangential penalty factor (3)	0
ktmult	tangential stiffness multiplier (3)	1
seg_up	nr of segment updates (4)	0

The slave and master surfaces 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.

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

Instead of quadrilateral surface elements (quad4) you may also use triangular elements (tri3).

### Comments:

1. For sliding interfaces the penalty can also be calculated automatically. To use this feature, add the *auto* attribute to the penalty element.

```
<penalty auto="on">1.0</penalty>
```

In this case, the specified value will scale the automatically calculated penalty factor.

2. Using the single-pass algorithm the contact tractions are calculate only on the slave surface. Although this is sufficient for most problems, this can also introduce a bias in the solution. For this reason it is possible to apply a two-pass algorithm. In this case, the single-pass algorithm is applied first, after which the roles of master and slave are switched and the single-pass algorithm is applied again.

3. As of version 1.2 FEBio can deal with friction. Three parameters control the frictional response: *fric\_coeff* is the material's friction coefficient and its value must be in the range from zero to one; *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 one, but it is observed that reducing this value might sometimes improves convergence.
4. 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 flipping segments is undesirable since it might cause instabilities in the solution process. 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 jump to new master segments. The default value is zero which means that FEBio will do a segment update each iteration of each timestep.

### 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.

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 in stead of quadrilateral elements (quad4).

### Tied Interfaces

A tied interface can be used to connect two non-conforming meshes. To define a tied interface you need to define 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>
```

Instead of quadrilateral surface elements (quad4) you may also use triangular elements (tria3).

### 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.

### 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.

### 3.6. *Globals* Section

In the *Globals* section you can define some global variables. Currently you can only define the body force here. 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.

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

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.

### 3.7. 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">
  <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.

FEBio also defines a “zero” loadcurve. This loadcurve is used for all time-dependent parameters that have no loadcurve specified (that is, the *lc* attribute is omitted). This loadcurve will have the effect that parameters are ramped up linearly from zero to their specified value. Imagine for instance, a rigid body defined as follows.

```
<material id="1" type="rigid body">
  <density>1.0</density>
  <trans_x type="prescribed">2.0</trans_x>
</material>
```

The *x* degree of freedom is prescribed but no loadcurve is specified. FEBio will automatically use the “zero” loadcurve and ramp up the value from 0 at the start time to 2 at the end time.

### 3.8. 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 (most of) the results. However, in some cases it might be useful to record additional results that are not stored by default. The user can request additional data by defining an *Output* section. FEBio allows the user to store this additional data in the logfile or the plotfile.

#### 3.8.1. Logfile

The user can request FEBio to output additional data in 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, the default name is used. Additional data is stored 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 mathematical expression using any valid combination of variable names and arithmetic operators. The available variable names depend on the data class and are defined below. A single *data* attribute can define multiple data expressions by separating them with a semicolon (;). For example, the data expression

```
data="x;y;z"
```

will store the variables *x*, *y* and *z* in separate columns. For more examples see below.

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 you will find the following entry 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.

### 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
<b>x</b>	x-coordinate of current nodal position
<b>y</b>	y-coordinate of current nodal position
<b>z</b>	z-coordinate of current nodal position
<b>ux</b>	x-coordinate of nodal displacement
<b>uy</b>	y-coordinate of nodal displacement
<b>uz</b>	z-coordinate of nodal displacement

For analyses using poroelastic materials, the following additional variables can be defined.

Node variables	Description
<b>p</b>	fluid pressure
<b>vx</b>	x-component of solid velocity
<b>vy</b>	y-component of solid velocity
<b>vz</b>	z-component of solid velocity

For example, to store the current nodal positions of all nodes, use the following *node\_data* element.

```
<node_data data="x;y;z"></node_data>
```

You can store the total nodal displacement for nodes 1 through 100, and all even numbered nodes 200 through 400 as follows.

```
<node_data data="sqrt(ux*ux+uy*uy+uz*uz)">1:100:1,200:400:2</node_data>
```

## 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
<b>sx</b>	xx-component of the Cauchy stress
<b>sy</b>	yy-component of the Cauchy stress
<b>sz</b>	zz-component of the Cauchy stress
<b>sxy</b>	xy-component of the Cauchy stress
<b>syz</b>	yz-component of the Cauchy stress
<b>sxz</b>	xz-component of the Cauchy stress
<b>Ex</b>	xx-component of the Green-Lagrange strain
<b>Ey</b>	yy-component of the Green-Lagrange strain
<b>Ez</b>	zz-component of the Green-Lagrange strain
<b>Exy</b>	xy-component of the Green-Lagrange strain
<b>Eyz</b>	yz-component of the Green-Lagrange strain
<b>Exz</b>	xz-component of the Green-Lagrange strain

For analyses using poroelastic materials, the following additional variables can be defined.

Element variables	Description
<b>wx</b>	x-component of fluid flux
<b>wy</b>	y-component of fluid flux
<b>wz</b>	z-component of fluid flux

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

```
<element_data data="sxx;syy;szz;sxy;syz;sxz" name="element stresses" >
</element_data>
```

To store the element pressure, define the following element for elements 1 to 100, and 200 to 300.

```
<element_data data="-0.3*(sxx+syy+szz)">1:100,200:300</element_data>
```

### 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
<b>x</b>	x-coordinate of center of mass
<b>y</b>	y-coordinate of center of mass
<b>z</b>	z-coordinate of center of mass
<b>qx</b>	x-component of rotation quaternion
<b>qy</b>	y-component of rotation quaternion
<b>qz</b>	z-component of rotation quaternion
<b>qw</b>	w-component of rotation quaternion
<b>Fx</b>	x-component of the rigid body reaction force
<b>Fy</b>	y-component of the rigid body reaction force
<b>Fz</b>	z-component of the rigid body reaction force
<b>Mx</b>	x-component of the rigid body reaction torque
<b>My</b>	y-component of the rigid body reaction torque
<b>Mz</b>	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>
```

### 3.8.2. Plotfile

By default, all the results are stored in a binary database, referred to as the *plotfile*. Currently, FEBio uses the LSDYNA database format for the plotfile. This format only offers limited 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>
```

## Chapter 4 - Restart Input file

As of version 1.1.6 FEBio will output a restart file for you. This means that 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 you to continue a previously terminated analysis. In addition, it allows you 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>
```

In this chapter we discuss 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.

### 4.1. 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>
```

### 4.2. 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

### 4.3. The LoadData Section

In the LoadData section you can redefine some or all 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 you wish to redefine.

### 4.4. 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 of.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<febio_restart version="1.0">
  <Archive>out.dmp</Archive>
</febio_restart>
```

## Chapter 5 - Multi-step Analysis

As of version 1.1.6 you can solve multi-step problems. A multi-step analysis is defined using multiple steps, where in each step you 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.

### 5.1. The Step section

The multi-step analysis feature introduces a new section to the input file. For each step you need to define a *step* section, preferably at the bottom of the input file. In this step section, you 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. More about the difference between local versus global parameters is explained below.

After the LoadData section, you can define the Step sections, as many as you need. In each Step section you 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.

### 5.1.1. 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.

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

### 5.1.2. Boundary Settings

The Boundary section that is defined before the first Step section defines the global boundary conditions. These conditions remain enforced during the entire simulation. The Boundary section defined in each Step section lists the boundary conditions that 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.

## 5.2. 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>

```

## Appendix A. Fixed Format Input

This chapter describes the fixed format for the FEBio input file. By “fixed”, we mean that each section of the input file must follow the order specified in this document. This also applies for each line of each section and for each entry on each line. In addition, every entry on each line has a fixed number of columns where its value may appear on the line. The user must pay careful attention to these restrictions, since FEBio may not be able to find all input mistakes related to incorrectly placed parameters. This format is identical to the format used by NIKE3D [11], developed and maintained by the Methods Development Group at Lawrence Livermore National Laboratory, in the sense that FEBio can read and run most NIKE3D input files, though FEBio does not support all NIKE3D features. Although our experience has shown that all FEBio input files using this format can be run with NIKE3D, we have not performed exhaustive testing. The input format for NIKE3D is detailed in the NIKE3D User’s Manual. In this manual we only describe the features supported by FEBio.

The fixed format for the input file is composed of a series of sections:

1. Control section
2. Material section
3. Nodal coordinates section
4. Element connectivity section
5. Rigid node and facet section
6. Contact section
7. Load curve section
8. Concentrated nodal force section
9. Pressure boundary section
10. Prescribed displacement section
11. Body force section

Of all these sections, only the first four sections will appear in every input file. The other sections are optional and only need to be specified if the corresponding entry in the control section is non-zero. Note that all sections must appear in the order they are described in this manual. The rest of this chapter describes the sections in detail.

Note that any line that contains an asterisk (“\*”) in the first column is interpreted as a comment and is ignored by FEBio.

## **A.1. Control Section**

The control section contains entries that describe the size of the problem as well as the solver parameters. It is composed of ten lines, and although the last three lines are ignored in FEBio, their presence is required for compatibility with NIKE3D.

### **A.1.1. Control Line 1**

<u>Columns</u>	<u>Format</u>
1 – 72	A256

heading to appear on output. (1)

#### *Comments.*

1. This line contains the problem title. In NIKE3D this line is limited to 72 characters, but in FEBio this line can be up to 256 characters. Note however that in FEBio (as well as in NIKE3D) only the first forty characters are saved to the plot file, a limitation that is imposed by the TAURUS file format used by NIKE3D and other LLNL codes including DYNA3D.

### A.1.2. Control Line 2

<u>Columns</u>		<u>Format</u>
1-2	Input format (1)	A2
3-5	Number of materials	I3
6-15	Number of node points	I10
16-25	Number of solid elements (2)	I10
26-35	<i>(not used)</i>	
36-45	Number of shell elements (3)	I10
46-50	<i>(not used)</i>	
51-55	Number of sliding interfaces	I5
56-65	<i>(not used)</i>	
66-70	Number of rigid nodes and facet lines	I5

#### *Comments*

1. The input format must be “FL”, which corresponds to NIKE3D version 3.0 and later.
2. This number includes all hexahedral (“brick”), pentahedral (“wedge”) and tetrahedral elements.
3. Currently FEBio only supports rigid shells. These are shells that are assigned to a rigid material. FEBio also ignores shell thicknesses.

### A.1.3. Control Line 3

<u>Columns</u>		<u>Format</u>
1-10	Number of time or load steps ( <i>ntime</i> )	I10
11-20	Time or load step size ( <i>dt</i> )	E10.0
21-25	Auto time/load step control flag (1) EQ. " " : use fixed time step size EQ. "AUTO" : enable standard automatic time step control	1x,A4
26-30	Maximum number of retries allowable per step (2) EQ.0: default = 5	I5
31-35	Optimal number of iterations per step (3) EQ.0: default = 11	I5
36-45	Minimum allowable step/load size EQ.0: default = $dt / 3$	E10.0
46-55	Maximum allowable step/load size EQ.0: default = $dt * 3$	E10.0

#### *Comments*

1. The automatic time stepping algorithm will adjust the time step size based on convergence information returned by the quasi-Newton solver. The time step size is bracketed by the minimum and maximum values that are input on this line. The algorithm also ensures that FEBio attempts to obtain a converged state at the desired termination time, determined by ( $ntime \times dt$ ).
2. This is the maximum number of attempts that the auto time stepper retries a failed quasi-Newton iteration. After each attempt the auto time stepper decreases the time step and restarts the iteration. When the minimum time step size is reached before the maximum number of iterations, the program ends in *error termination*.
3. The optimal number of iterations is the expected average number of iterations for each time step. By comparing this number with the actual number of iterations, the auto time stepper decides whether to increase or decrease the time step size.

### A.1.4. Control Line 4

<u>Columns</u>		<u>Format</u>
1-5	Number of load curves (1)	I5
6-10	Maximum number of points defining any load curve (2)	I5
11-15	Number of concentrated nodal loads	I5
16-20	Number of element surfaces with applied pressure loading	I5
21-25	Number of displacement boundary conditions	I5
26-35	<i>(not used)</i>	
36-40	Body force loads due to base acceleration in x-direction EQ.0: no x-acceleration NE.0: x-acceleration	I5
41-45	Body force loads due to base acceleration in y-direction EQ.0: no y-acceleration NE.0: y-acceleration	I5
46-50	Body force loads due to base acceleration in z-direction EQ.0: no z-acceleration NE.0: z-acceleration	I5

#### *Comments*

1. A *loadcurve* is simply a list of data pairs that represents a curve of time versus load. The load can be interpreted in different ways. For example, it might represent a rigid body displacement, a concentrated nodal force, or any other variable that may depend on time.
2. This number is ignored in the current version of FEBio.

### A.1.5. Control Line 5

<u>Columns</u>		<u>Format</u>
1-35	( <i>not used</i> )	
36-40	Dump file creation flag (1) EQ.0: no dump file will be created EQ.1: dump file will be created	I5

#### *Comments*

1. The dump file allows the user to restart an unfinished problem at a later time. When the flag is set to 1, FEBio will create a dump file (called *f3dump*) that can be used to restart the analysis from the last converged time step.

### A.1.6. Control Line 6

<u>Columns</u>		<u>Format</u>
1-30	( <i>not used</i> )	
31-35	Maximum number of Quasi-Newton (1) equilibrium iterations permitted between stiffness matrix reformations. EQ.0: default set to 10	I5
36-40	Maximum number of stiffness matrix reformations per time step (2) EQ.0: default set to 15	I5
41-50	Convergence tolerance on displacements EQ.0: default set to 0.001	E10.0
51-60	Convergence tolerance on energy EQ.0: default set to 0.01	E10.0
61-70	Convergence tolerance on residual EQ.0: default set to 1E+10 (i.e. deactivated)	E10.0
71-80	Convergence tolerance on line search EQ.0: default set to 0.9	E10.0

#### *Comments*

1. FEBio uses the BFGS method [12] to solve the nonlinear finite element equations. In this method the stiffness matrix does not get recalculated during every iteration (such as in the Full-Newton method). In stead it calculates an approximation of the (inverse of the) stiffness matrix, which is called a *stiffness update*. The second parameter on this control line sets the maximum number of such updates (one during every iteration) before the exact stiffness matrix is reformed.
2. This parameter sets the maximum number of stiffness reformations per time step. If this number is reached the time step fails and, if the auto-time stepper is activated, a smaller time step will be tried.

**A.1.7. Control Line 7**Columns

1-5

analysis type

EQ. 0: quasi-static analysis

Format

I5

**A.1.8. Control Line 8**

This line is not used in FEBio.

**A.1.9. Control Line 9**

This line is not used in FEBio.

**A.1.10. Control Line 10**

This line is not used in FEBio.

## A.2. Material section

The *Material Section* contains all the material parameters of all the materials used in the problem. Repeat the following set of eight lines for each material. If the material is used for shell elements then an additional two lines are expected. Although FEBio reads those lines in, it currently ignores the values. The total number of materials is entered on control line 2. For detailed descriptions of the materials, please see Section 3.3 above.

### A.2.1. Line 1

<u>Columns</u>	<u>Format</u>
1-5	I5
6-10	I5
11-20	E10.0
21-25	I5

---

Material identification number	
Material Type	
EQ.1: neo-Hookean material	
EQ.15: Mooney-Rivlin	
EQ.18: Transversely isotropic Mooney-Rivlin	
EQ.20: Rigid body	
Density	
Element class for which this material is used	
EQ.20: brick element (also pentahedral and tetrahedral element)	
EQ.2: shell element	

### A.2.2. Line 2

<u>Columns</u>	<u>Format</u>
1-72	

---

Material title	
----------------	--

### A.2.3. Material Type 1 – Neo-Hookean

<u>Columns</u>			<u>Format</u>
1-10	<b>Line 3</b>	Young's modulus	E10.0
1-10	<b>Line 4</b>	Poisson's ratio	E10.0
	<b>Line 5-8</b>	blank	

### A.2.4. Material Type 15 – Mooney Rivlin Hyperelastic

<u>Columns</u>			<u>Format</u>
1-10	<b>Line 3</b>	Coefficient of first invariant $A$	E10.0
1-10	<b>Line 4</b>	Coefficient of second invariant $B$	E10.0
1-10	<b>Line 5</b>	Poisson's ratio $\nu$	E10.0
	<b>Line 6-8</b>	Blank	

### A.2.5. Material Type 18 – Transversely Isotropic Hyperelastic

<u>Columns</u>			<u>Format</u>
1-10	<b>Line 3</b>	Isotropic material coefficient $C_1$	E10.0
11-20		Isotropic material coefficient $C_2$	E10.0
21-30		Exponential stress coefficient $C_3$	E10.0
31-40		Fiber uncrimping coefficient $C_4$	E10.0
41-50		Modulus of straightened fibers $C_5$	E10.0
1-10	<b>Line 4</b>	Bulk modulus $K$	E10.0
11-20		Fiber stretch for straightened fibers $\lambda_m$	E10.0
1-10	<b>Line 5</b>	<i>(not used)</i>	
1-10	<b>Line 6</b>	Material axes option, $AOPT$ Fiber axis is always aligned with load axis $a$	E10.0
		EQ.0: local material axes given by local element Nodes specified on line 7 below. Line 8 is blank	
		EQ.1: local material axes determined by a point in space and global location of each element integration point. Line 8 is blank.	

		EQ.2: local material axes determined by normalized vector <b>a</b> .	
1-30	<b>Line 7</b>	<i>AOPT</i> .EQ.0: local element nodes (default=1,2,4)	3E10.0
		<i>AOPT</i> .EQ.1: $x_p, y_p, z_p$	3E10.0
		<i>AOPT</i> .EQ.2: $a_1, a_2, a_3$	3E10.0
1-30	<b>Line 8</b>	(not used)	3E10.0
31-40		AC – (active contraction) load curve number	E10.0
41-50		AC – intracellular calcium concentration	E10.0
51-60		AC – tension-sarcomere length relation constant	E10.0
61-70		AC – Unloaded sarcomere length	E10.0
71-80		AC – no tension sarcomere length	E10.0

The local fiber direction is specified using the *AOPT* parameter. For *AOPT*=0 the material axis is defined by the local element node numbering specified on line 7. For *AOPT*=1 the material axis is defined a fixed point in space and the global position of the element Gauss points. The normalized vector connecting these two points defines the axis. For *AOPT*=2 the material axis is defined by a global vector specified on line 7.

## A.2.6. Material type 20 – Rigid Body

<u>Columns</u>		<u>Format</u>
	<b>Line 3</b> Blank	
	<b>Line 4</b> Blank	
1-10	<b>Line 5</b> X-translational boundary code	E10.0
11-20	Y-translational boundary code	E10.0
21-30	Z-translational boundary code	E10.0
31-40	X-rotational boundary code	E10.0
41-50	Y-rotational boundary code	E10.0
51-60	Z-rotational boundary code	E10.0
1-10	<b>Line 6</b> Center of mass input flag NE.1: FEBio computes center of mass EQ.1: read user supplied coordinates below	E10.0
11-20	X – coordinate of center of mass	E10.0
21-30	Y – coordinate of center of mass	E10.0
31-40	Z – coordinate of center of mass	E10.0
	<b>Line 7-8</b> Blank	

Nodal constraints and displacement boundary conditions on rigid body *nodes* are ignored. Instead, constraints are applied at the rigid body center of mass. The coordinates of the center of mass are either computed by FEBio or may be input by the user. The boundary condition codes are interpreted as:

*code* < 0: fixed  
*code* = 0: free  
*code* > 0: number of load curve for prescribed displacement or rotation (in radians).

### **A.3. Nodal coordinates section**

The following line is repeated for every node of the mesh. The total number of nodes is entered on line 2 of the Control section.

<u>Columns</u>		<u>Format</u>
1-8	Node number	I8
9-13	Displacement boundary condition code EQ.0: no constraints EQ.1: constrained $x$ displacement EQ.2: constrained $y$ displacement EQ.3: constrained $z$ displacement EQ.4: constrained $x$ and $y$ displacement EQ.5: constrained $y$ and $z$ displacement EQ.6: constrained $z$ and $x$ displacement EQ.7: constrained $x$ , $y$ and $z$ displacement	I5
14-33	$x$ – coordinate	E20.0
34-53	$y$ – coordinate	E20.0
54-73	$z$ – coordinate	E20.0
74-78	(ignored)	I5

#### A.4. Element connectivity section

The following line is repeated for every element in the mesh. The total number of elements is entered on line 2 of the Control section.

<u>Columns</u>		<u>Format</u>
1-8	Element number	I8
9-13	Material number	I5
14-21	Node point 1	I8
22-29	Node point 2	I8
...	...	
70-77	Node point 8	I8

Nodes 1 through 8 define the corner nodes of the 8-node hexahedral or “brick” element. In the case of a pentahedral or “wedge” element the sixth node is repeated for entry 7 and 8. In the case of a tetrahedral element, the fourth node is repeated four times.

- hex :  $n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8$
- penta :  $n_1, n_2, n_3, n_4, n_5, n_6, n_6, n_6$
- tet :  $n_1, n_2, n_3, n_4, n_4, n_4, n_4, n_4$

### **A.5. Rigid node and facet section**

The number of rigid node and facet lines is defined in the control section on line 2.

<u>Columns</u>		<u>Format</u>
1-5	Rigid body number	I5
6-13	Node point 1	I8
14-21	Node point 2	I8
22-29	Node point 3	I8
30-37	Node point 4	I8

Rigid nodes and facets move as part of the indicated rigid body. The input is a very general list of node points, four (or less) per line. The nodes need not be unique and need not form a facet of an element, although this method of input is often convenient. This feature gives a convenient way to attach a portion of a deformable mesh to a rigid body without merging node points or using a tied interface. Note that the nodes in this section must be part of a deformable mesh; this section cannot be used to define new geometry.

## A.6. Contact section

Contact between objects is defined via sliding interface surfaces. The number of sliding interfaces is defined on control line 1 of the control section. Each sliding interface consists of a master surface and a slave surface. When using a two-pass algorithm for enforcement of the contact constraint (REF), the definition of master and slave surfaces is arbitrary. 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 master. The master and slave contact surfaces are described by “facets” of existing elements in the model. A facet is defined by the node numbers that construct either a quadrilateral or triangular polygon. For each sliding interface the following lines are repeated.

### A.6.1. Control line

<u>Columns</u>		<u>Format</u>
1-8	Number of slave facets	I8
9-16	Number of master facets	I8
17-20	Interface type EQ.3: two-pass algorithm EQ.-3: single-pass algorithm	I4
21-30	Penalty factor (1)	E10.0
31-75	<i>(not used)</i>	
76-80	Auxiliary interface control line flag (IAUG) EQ.1: read an auxiliary interface control line flag immediately following this line.	I5

#### *Comments*

1. The penalty factor must be entered as a negative number to be consistent with Nike3D, since in Nike3D a positive number is interpreted as a scale factor to an automatically calculated penalty factor. If a negative number is entered, the absolute value is used as the actual penalty factor. In FEBio the absolute value of this number is always used as the actual penalty factor.

### A.6.2. Auxiliary control line

Add this control line immediately after the control line if IAUG equal 1.

<u>Columns</u>		<u>Format</u>
1-5	<i>(not used)</i>	
6-15	Normal direction convergence tolerance for augmentations (ALTOL) GT.0: converged when force norm < ALTOL EQ.0: DEFAULT = 0.1	E10.0

### A.6.3. Slave facet cards

For each slave facet repeat the following line. There must be at least three non-repeated node numbers but no more than four node numbers per line. Triangular facets are entered by repeating the last node.

<u>Columns</u>	<u>Format</u>
1-8	slave facet number
9-16	node point n1
17-24	node point n2
25-32	node point n3
33-40	node point n4

### A.6.4. Master facet cards

The format for entering master facets is the same as for the slave facets. For each master facet repeat the following line.

<u>Columns</u>	<u>Format</u>
1-8	slave facet number
9-16	node point n1
17-24	node point n2
25-32	node point n3
33-40	node point n4

If there are multiple sliding interfaces, the order of the cards is as follows.

```
Control line 1 - sliding interface 1
Control line 2 - sliding interface 1 (optional)
...
Control line 1 - sliding interface n
Control line 2 - sliding interface n (optional)
Slave surface - sliding interface 1
Master surface - Sliding interface 1
...
Slave surface - sliding interface n
Master surface - Sliding interface n
```

## A.7. Load curve section

A load curve is a list of data pairs that represent a discretized function of time. They are used to specify the magnitude and/or time variation of many types of data, including boundary conditions, material properties and motion of rigid materials. Any boundary condition or property may reference any load curve, and a load curve may be referenced more than once.

Define the number of load curves on Control line 4. Repeat the following lines for every load curve:

### A.7.1. Line 1

<u>Columns</u>		<u>Format</u>
1-5	Load curve number	I5
6-10	Number of points in load curve (pts)	I5

### A.7.2. Line 2,...,pts

<u>Columns</u>		<u>Format</u>
1-10	Time	E10.0
11-20	Load curve value	E10.0

The value of a time point in between two load points is calculated using linear interpolation. The value of a time point outside the domain of the load curve is clamped to the nearest data point.

### **A.8. Concentrated nodal force section**

Define the number of concentrated nodal loads as specified on control line 4. Repeat the following line for every nodal load.

<u>Columns</u>		<u>Format</u>
1-8	Node point number on which this load acts	I8
9-13	Direction in which load acts EQ.1: $x$ – direction force EQ.2: $y$ – direction force EQ.3: $z$ – direction force	I5
14-18	Load curve number	I5
19-28	Scale factor EQ.0: default set to “1.0”	E10.0

Concentrated nodal loads are defined in global coordinates. In contrast to “follower forces”, concentrated nodal loads act in a fixed direction as over the course of the analysis. Hence, loads that were defined normal to a surface in the undeformed configuration may not remain normal if the surface undergoes large deformation. Pressure loads may be used to maintain normality during large deformation.

### **A.9. Pressure boundary section**

Define the number of lines as specified on control card 4. Repeat the following line for every pressure load. For triangular facets repeat the third point.

<u>Columns</u>	<u>Format</u>
1-5	I5
6-13	I8
14-21	I8
22-29	I8
30-37	I8
38-47	E10.0
48-57	E10.0
58-67	E10.0
68-77	E10.0

Pressure forces are “follower forces”. They always remain normal to the surface even under large deformations. The magnitude of the pressure is determined by the load curve value and the scale factor. When a scale factor is zero, it is *not* replaced by the default value of 1.0. Only when all four scale factors are zero, then they are replaced with the default values.

### ***A.10. Prescribed displacement section***

Define the number of lines as specified on control line 4.

<u>Columns</u>		<u>Format</u>
1-8	Node point number to which this node is applied	I8
9-13	Global direction in which node is displaced EQ.1: translation in $x$ – direction EQ.2: translation in $y$ – direction EQ.3: translation in $z$ – direction	I5
14-18	Load curve number	I5
19-28	Scale factor on load curve EQ.0: default set to “1.0”	E10.0

### A.11. Body force section

The body force due to base acceleration is specified on Control card 4. Skip lines 1, 2, and/or 3 if the corresponding flag is zero. The sign convention for body force loads is understood by considering that the coordinate system, or base, is accelerated in the direction specified. Thus, a structure with its base fixed in the X-Y plane and extending upward in the +Z direction will be crushed by a +Z acceleration, and stretched by a -Z acceleration. Material mass density must be specified for all analyses using body forces.

#### A.11.1. Line 1

<u>Columns</u>		<u>Format</u>
1-5	Load curve number	I5
6-15	Scale factor on X acceleration EQ.0: default set to "1.0"	E10.0

#### A.11.2. Line 2

<u>Columns</u>		<u>Format</u>
1-5	Load curve number	I5
6-15	Scale factor on Y acceleration EQ.0: default set to "1.0"	E10.0

#### A.11.3. Line 3

<u>Columns</u>		<u>Format</u>
1-5	Load curve number	I5
6-15	Scale factor on Z acceleration EQ.0: default set to "1.0"	E10.0

## Appendix B. 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 2.2 for more information on how to use this file. In this section we detail 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, PSLDLT, Skyline .... Not all solvers are available for all platforms. Only the Skyline solver is available for all platforms and as of version 1.2 the SuperLU and the Pardiso solver are also available on most platforms.

## References

- [1] Bonet, J., and Wood, R. D., 1997, *Nonlinear continuum mechanics for finite element analysis*, Cambridge University Press.
- [2] 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.
- [3] Veronda, D. R., and Westmann, R. A., 1970, "Mechanical Characterization of Skin - Finite Deformations," *J. Biomechanics*, Vol. 3, pp. 111-124.
- [4] 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.
- [5] Quapp, K. M., and Weiss, J. A., 1998, "Material characterization of human medial collateral ligament," *J Biomech Eng*, 120(6), pp. 757-763.
- [6] 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.
- [7] 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," *J Biomech Eng*, 102(1), pp. 73-84.
- [8] 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.
- [9] Wayne, J. S., Woo, S. L., and Kwan, M. K., 1991, "Application of the u-p finite element method to the study of articular cartilage," *J Biomech Eng*, 113(4), pp. 397-403.
- [10] 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.
- [11] 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.
- [12] Matthies, H., and Strang, G., 1979, "The solution of nonlinear finite element equations," *Intl J Num Meth Eng*, 14, pp. 1613-1626.