Version 1.0

User's Manual

Written and Edited by

Steve Maas and 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

jeff.weiss@utah.edu

Last Updated: August 29, 2007

Table of Contents

CHAPTER 1. Introduction

1.1. Overview of FEBio

FEBio is a nonlinear finite element solver that is specifically designed for biomechanical applications. It offers modeling scenarios, constitutive models and boundary conditions that are relevant to 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 *quasi-static-poroelastic*. In a *quasi-static* analysis the (quasi-) static response of the system is sought; inertial terms are ignored. In a *quasi-static-poroelastic* analysis a coupled solid-fluid problem is solved. This analysis type is useful for modeling tissues that have a mobile fluid phase and the accurate modeling of the fluid behavior is important.

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 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 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. 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 files can be imported in PreView and can be exported as a FEBio input file.

1.2. About this document

This document is a 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 user 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 formats 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. FEBio supports two input formats. Chapter 3 describes the *free* input format. This input format is an XML-based format that organizes the data in a convenient hierarchical structure. In chapter 4 the details of the *fixed* format input file are described. This format lists the problem data in a predefined sequential list. The latter format is almost identical to the input specification of NIKE3D. Therefore FEBio can run NIKE3D input files, however with some limitations as discussed in chapter 4.

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 Linux. 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 fixed format
- `-r` : restart file name
- `-g[n]` : debug flag (does not require a file name)
- `-p`: plot file name
- `-d`: dump file name
- `-o`: log file name

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 ".n" extension or no extension indicates to FEBio that the input file is formatted in the fixed format, while the extension ".feb" or ".xml" implies that the input file is formatted in the free format.

Example: > `febio -i input.n`

The `-r` option allows you to restart a previously started analysis. The filename that must follow this option is the name of a binary "*dump*" file that was created by a previous call to FEBio. The `-i` and `-r` options are mutually exclusive; only one of them may appear on the command line.

Example: > `febio -r f3dmp`

The `–g[n]` option runs FEBio in *debug mode*. The *n* must be 0, 1 or 2. See the next section for more information on running FEBio in debug mode.

Example: > `febio -i input.n –g2`

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 that was written to the screen. The *plot file* contains the results of the analysis. Since this file is binary, the results must be analyzed using postprocessing 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. The names of these files can be specified with the command options `-p` (plot file), `-d` (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 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. 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*.

2.2. Advanced Options

Interrupting a run

The user can pause the run by pressing *ctrl+z*. This will bring up a menu, and the user can pick from several options. The user can enter his choice by typing one of the following commands on the command line.

- *cont*: continue run without change
- *conv*: force the current time step to converge
- *help*: list the available commands with a short explanation
- *plot*: dump current state to plot database and continue
- *quit*: exit the application

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

Debugging a run

As stated in section 2.1 you can run FEBio in debug-mode by specifying the `-g[n]` option on the command line. Three debug levels are available.

- -g0: debug mode is off. No information is saved to the log file and only the converged states are written to the plot file.
- -g1: write quasi-Newton convergence information to screen and file and save converged states to file. (default mode)
- -g2: same as –g1 but also save plot state for every quasi-Newton iteration.

The last option is useful for debugging an input file that passes the syntax check but does not run to completion (e.g., “error termination”). When no option is entered on the command line, the default option is –g1.

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 (ADD REF). 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.
- *Boundary*: Defines the boundary conditions that are applied on the geometry.
- *LoadData*: Defines the load curve data.

Example input files are provided in section 3.7.

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 that *must* be specified and optional parameters.

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.
`<step_size>`	The initial time step size. (= *dt*)

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	(off)
`<max_refs>`	Max number of stiffness reformations	15
`<max_ups>`	Max number of BFGS stiffness updates	10

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 number of retries allowed per time step	5
`<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.

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*: isotropic linear elastic
- *St-Venant-Kirchhoff*: St Venant-Kirchhoff elastic
- *neo-Hookean*: Neo-Hookean hyperelastic
- *Mooney-Rivlin*: Mooney-Rivlin hyperelastic
- *Veronda-Westmann*: Veronda-Westmann hyperelastic
- *trans iso Mooney-Rivlin*: transversly isotropic Mooney-Rivlin
- *trans iso Veronda-Westmann*: transversely isotropic Veronda-Westmann
- *rigid body*: rigid body material
- *poroelastic*: poroelastic material

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*.

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,

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

Here, λ and μ are the Lamé parameters and are related to the more familiar Young's modulus E and Poisson's ratio ν as follows,

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

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

This material model is only valid for small strains and small rotations. It is often convenient to express the material properties with the bulk modulus K and shear modulus G. To convert to Young's modulus and Poisson's ratio you can use the following formulas.

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

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

Example:

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

St. Venant-Kirchhoff

The material type for a St. Venant-Kirchhoff material is S*t. 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

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

Here, λ and μ are the Lamé parameters and are related to the more familiar Young's modulus E and Poisson's ratio ν as follows,

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

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

Example:

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

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 **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>
```

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,

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

C_1 and C_2 are the Mooney-Rivlin material coefficients. The variables $\tilde{I}_1$ and $\tilde{I}_2$ are the first and second invariants of the deviatoric right Cauchy-Green deformation tensor $\tilde{\mathbf{C}}$. The coefficient *K* is a bulk modulus-like penalty parameter and *J* is the determinant of the deformation gradient tensor. When $C_2 = 0$, this model reduces to an uncoupled version of the neo-Hookean constitutive model.

This material model uses a three-field element formulation, interpolating displacements as linear field variables and pressure and volume ratio as piecewise constant on each element [2].

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

Example:

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

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:

$$W = C_1\left[e^{\left(C_2\left(\tilde{I}_1-3\right)\right)}-1\right]-\frac{C_1C_2}{2}\left(\tilde{I}_2-3\right)+U\left(J\right)$$

The dilatational term is identical to the on 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>
```

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:

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

Here $\tilde{I}_1$ and $\tilde{I}_2$ are the first and second invariants of the deviatoric version of the right Cauchy Green deformation tensor $\tilde{\boldsymbol{C}}$ and $\tilde{\lambda}$ is the deviatoric part of the stretch along the fiber direction ($\tilde{\lambda}^2 = \boldsymbol{a}_0 \cdot \tilde{\boldsymbol{C}} \cdot \boldsymbol{a}_0$, where $\boldsymbol{a}_0$ is the initial fiber direction), and $J = \det\left(\boldsymbol{F}\right)$ is the Jacobian of the deformation (volume ratio). The function F_1 represents the material response of the isotropic ground substance matrix and is the same as the Mooney-Rivlin form specified above, while F_2 represents the contribution from the fiber family. The strain energy of the fiber family is as follows:

$$\tilde{\lambda}\frac{\partial F_2}{\partial \tilde{\lambda}} = 0, \quad \tilde{\lambda} \le 1$$

$$\tilde{\lambda}\frac{\partial F_2}{\partial \tilde{\lambda}} = C_3\left(e^{C_4\left(\tilde{\lambda}-1\right)} - 1\right), \quad 1 < \tilde{\lambda} < \lambda_m$$

$$\tilde{\lambda}\frac{\partial F_2}{\partial \tilde{\lambda}} = C_5 + C_6\tilde{\lambda}, \quad \tilde{\lambda} \ge \lambda_m$$

Here, C_1 and C_2 are the Mooney-Rivlin material coefficients, *lam_max* (λ_m) is the stretch at which the fibers are straightened, C_3 scales the exponential stresses, C_4 is the rate of uncrimping of the fibers, and C_5 is the modulus of the straightened fibers. C_6 is determined from the requirement that the stress is continuous at λ_m.

This material model uses a three-field element formulation, interpolating displacements as linear field variables and pressure and volume ratio as piecewise constant on each element [2].

The element fiber direction is specified using the *fiber* parameter. This parameter has an attribute named *type* that can take the following values.

- *local*: material axis determined by local element nodes. The value is interpreted as the local node numbers of the nodes that define the direction of the axis. The following example defines a local fiber axis by local element nodes 1 and 2.

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

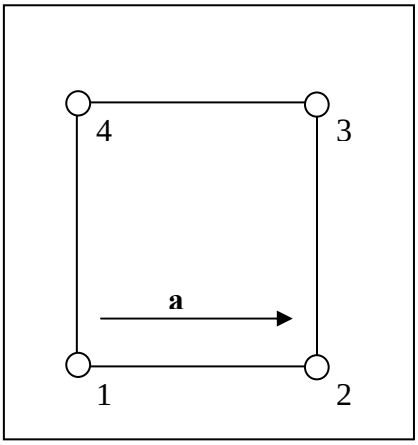

Figure 3-1. *local* fiber direction option .

- *spherical*: material axis 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>
```

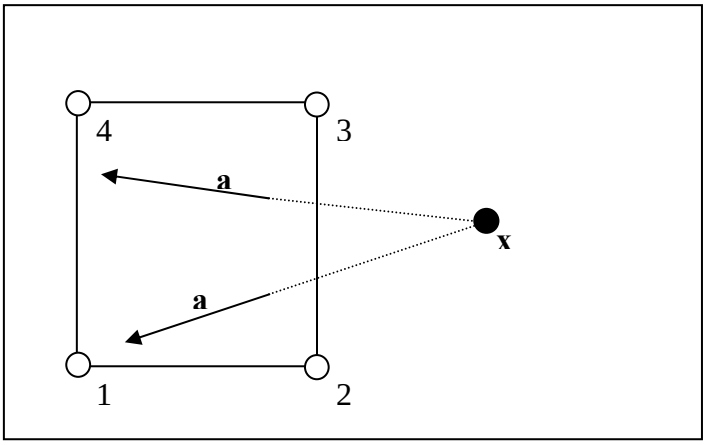

Figure 3-2. *spherical* fiber direction option.

- *vector*: material axis is specified by vector. The value is the direction of the material axis. The following defines all element fiber directions in the direction of the vector [1,0,0].

```
<fiber type="vector">1,0,0</fiber>
```

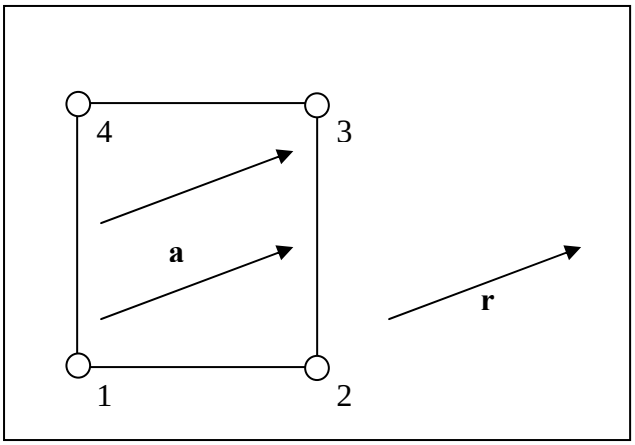

Figure 3-3. *vector* fiber direction option.

If the *type* attribute is omitted the fiber distribution follows the local element nodes 1 and 2. So, this would be the same as adding a fiber parameter such as the example above under *local*.

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>
      <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>2.04</l0>
            <refl>1.58</refl>
      </active_contration>
</material>
```

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 the same as the Mooney-Rivlin transversely isotropic model. 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>
```

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>
```

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 *type* specifies the type of the translation.
<trans_y>	y-translation code. Attribute *type* specifies the type of the translation.
<trans_z>	z-translation code. Attribute *type* specifies the type of the translation.
<rot_x>	x-rotation code. Attribute *type* specifies the type of the rotation.
<rot_y>	y-rotation code. Attribute *type* specifies the type of the rotation.
<rot_z>	z-rotation code. Attribute *type* specifies the type of the rotation.

If the *center_of_mass* parameter is omitted then FEBio will calculate the center of mass automatically. In this case you *must* specify a density. 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>
```

Here, the type can be either *free, fixed* or *prescribed*. If the type is prescribed then the *lc* attribute can be used to specify a load curve. 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.

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.

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

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 *id*s 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).

Elements Section

The element section contains all the element connectivity data. Different element types have different tags. The following solid elements are available in FEBio.

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. And second a material number. This number is the material *id* that was defined in the material section.

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; currently only rigid shell elements are supported in 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 and rigid joints.

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="l"]>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.

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="l"]>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

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.

Pressure forces

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

```
<pressure>
      <quad4 id="n" [lc="l" 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.

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:

- *sliding_with_gaps*: defines a sliding interface that may separate
- *rigid*: defines a rigid interface
- *rigid_joint*: defines a joint between two rigid bodies

Sliding Interfaces

This type of contact interface demands the declaration of a *slave* and *master* surface, in addition to some control parameters. The control parameters are given by.

`tolerance`	augmentation tolerance	0.01

penalty	penalty factor	1.0

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).

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. 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.

3.7. Example Files

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- This is an example of the free format input for FEBio -->
<febio_spec version="0.1">
	<Control>
		<time_steps>10</time_steps>
		<step_size>0.1</step_size>
	</Control>
	<Material>
		<material id="1" type="neo-Hookean">
			<E>100.0</E>
			<v>0.45</v>
		</material>
	</Material>
	<Geometry>
		<Nodes>
			<node id="1">0,0,0</node>
			<node id="2">1,0,0</node>
			<node id="3">1,1,0</node>
			<node id="4">0,1,0</node>
			<node id="5">0,0,1</node>
			<node id="6">1,0,1</node>
			<node id="7">1,1,1</node>
			<node id="8">0,1,1</node>
		</Nodes>
		<Elements>
			<hex8 id="1" mat="1">1,2,3,4,5,6,7,8</hex8>
		</Elements>
	</Geometry>
	<Boundary>
		<fix>
			<node id=”1” bc=”xyz”></node>
			<node id=”4” bc=”xyz”></node>
			<node id=”5” bc=”xyz”></node>
			<node id=”8” bc=”xyz”></node>
		</fix>
		<prescribe>
			<node id="2" bc="x">1.0</node>
			<node id="3" bc="x">1.0</node>
			<node id="6" bc="x">1.0</node>
			<node id="7" bc="x">1.0</node>
		</prescribe>
	</Boundary>
</febio_spec>
```

CHAPTER 4. 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.

4.1. 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.

Control Line 1

Colums		Format
1 – 72	heading to appear on output. (1)	A256

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.

Control Line 2

Colums		Format
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	(*not used*)	
36-45	Number of shell elements (3)	I10
46-50	(*not used*)	
51-55	Number of sliding interfaces	I5
56-65	(*not used*)	
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.

Control Line 3

Colums		Format
1-10	Number of time or load steps (*ntime*)	I10
11-20	Time or load step size (*dt*)	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* × *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.

Control Line 4

Colums		Format
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	(*not used*)	
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.

Control Line 5

Colums		Format
1-35	(*not used*)	
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.

Control Line 6

Colums		Format
1-30	(*not used*)	
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.

Control Line 7

Colums		Format
1-5	analysis type EQ. 0: quasi-static analysis	I5

Control Line 8

This line is not used in FEBio.

Control Line 9

This line is not used in FEBio.

Control Line 10

This line is not used in FEBio.

4.2. 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.

Line 1

Colums		Format
1-5	Material identification number	I5
6-10	Material Type EQ.1: neo-Hookean material EQ.15: Mooney-Rivlin EQ.18: Transversely isotropic Mooney-Rivlin EQ.20: Rigid body	I5
11-20	Density	E10.0
21-25	Element class for which this material is used EQ.20: brick element (also pentahedral and tetrahedral element) EQ.2: shell element	I5

Line 2

Colums		Format
1-72	Material title	

Material Type 1 – Neo-Hookean

Colums			Format
1-10	**Line 3**	Young's modulus	E10.0
1-10	**Line 4**	Poisson's ratio	E10.0
	Line 5-8	blank	

Material Type 15 – Mooney Rivlin Hyperelastic

Colums			Format
1-10	**Line 3**	Coefficient of first invariant *A*	E10.0
1-10	**Line 4**	Coefficient of second invariant *B*	E10.0
1-10	**Line 5**	Poisson's ratio ν	E10.0
	Line 6-8	Blank	

Material Type 18 – Transversely Isotropic Hyperelastic

Colums			Format
1-10	**Line 3**	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	**Line 4**	Bulk modulus *K*	E10.0
11-20		Fiber stretch for straightened fibers λ_m	E10.0
1-10	**Line 5**	(*not used*)	
1-10	**Line 6**	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 **a**.	
1-30	**Line 7**	*AOPT*.EQ.0: local element nodes (default=1,2,4)	3E10.0
		AOPT.EQ.1: x_p, y_p, z_p	3E10.0
		AOPT.EQ.2: a_1, a_2, a_3	3E10.0
1-30	**Line 8**	(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.

Material type 20 – Rigid Body

Colums			Format
	Line 3	Blank	
	Line 4	Blank	
1-10	**Line 5**	*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	**Line 6**	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
	Line 7-8	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).

4.3. 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.

Colums		Format
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

4.4. 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.

Colums		Format
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 : $n_1, n_2, n_3, n_4, n_4, n_4, n_4, n_4$

4.5. Rigid node and facet section

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

Colums		Format
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.

4.6. 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.

Control line

Colums		Format
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	(*not used*)	
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.

Auxiliary control line

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

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

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.

Colums		Format
1-8	slave facet number	I8
9-16	node point n1	I8
17-24	node point n2	I8
25-32	node point n3	I8
33-40	node point n4	I8

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.

Colums		Format
1-8	slave facet number	I8
9-16	node point n1	I8
17-24	node point n2	I8
25-32	node point n3	I8
33-40	node point n4	I8

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
```

4.7. 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:

Line 1

Colums		Format
1-5	Load curve number	I5
6-10	Number of points in load curve (pts)	I5

Line 2,…,*pts*

Colums		Format
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.

4.8. 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.

Colums		Format
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.

4.9. 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.

Colums		Format
1-5	Load curve 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
38-47	Scale factor at node 1	E10.0
48-57	Scale factor at node 2	E10.0
58-67	Scale factor at node 3	E10.0
68-77	Scale factor at node 4	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.

4.10. Prescribed displacement section

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

Colums		Format
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

4.11. 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.

Line 1

Colums		Format
1-5	Load curve number	I5
6-15	Scale factor on *X* acceleration EQ.0: default set to "1.0"	E10.0

Line 2

Colums		Format
1-5	Load curve number	I5
6-15	Scale factor on *Y* acceleration EQ.0: default set to "1.0"	E10.0

Line 3

Colums		Format
1-5	Load curve number	I5
6-15	Scale factor on *Z* acceleration EQ.0: default set to "1.0"	E10.0

CHAPTER 5. 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.