FEBioChem User Manual

Version 1.0

Table of Contents

[1 Introduction 1](#_Toc522176509)

[1.1 Mixtures 2](#_Toc522176510)

[2 Using the FEBioChem plugin 2](#_Toc522176511)

[3 Defining a FEBioChem model 3](#_Toc522176512)

[3.1 Module 3](#_Toc522176513)

[3.2 Globals 3](#_Toc522176514)

[3.3 Material 3](#_Toc522176515)

[3.3.1 species property 4](#_Toc522176516)

[3.3.2 solid\_bound\_species property 4](#_Toc522176517)

[3.3.3 reaction property 4](#_Toc522176518)

[3.4 Initial Values 5](#_Toc522176519)

[3.5 Boundary Conditions 5](#_Toc522176520)

[3.6 Prescribing a concentration flux 5](#_Toc522176521)

[3.7 Plot Variables 6](#_Toc522176522)

[3.8 Initial velocity 6](#_Toc522176523)

# Introduction

The FEBioChem plugin was designed for solving the reaction-diffusion equation in the context of a (non-deformable) mixture with multiple chemical species. The species can be solutes that flow with the solvent, or can be bound to the solid phase of the mixture. In this regard, this plugin offers similar capabilities as the FEBioMix module in FEBio (at least for non-deformable mixtures). However, since the mixture is assumed non-deformable, the calculations are optimized and the performance improved.

The basic set of equations that FEBioChem solves are the nonlinear reaction-diffusion equations, which can be stated in the following form.



Here,  represents the concentration of chemical species *i,*  the diffusion constant of species *i*, and  accounts for all local reactions that contribute to the concentration of species *i*. The vector  represents all the concentrations, and **v** represents a velocity field in which the solutes can flow.

A general (forward) chemical reaction is notated as follows.



Here, *Mi* identifies the chemical species, and  are the stoichiometric coefficients of the reactants and products, respectively, *kj* is the rate coefficient.

The source term takes on the following form,



Here, the sum is over *J* reactions and  and,



the reaction rate of reaction *j*. Usually, the exponent is taken to be , as according to the law of mass action.

## Mixtures

Thus far, we considered the case where the solute is free, i.e. not inside a porous material. In the case if the solutes are contained inside a porous material, then the equations above remain valid, except now that the concentrations are with respect to the mixture volume. The ‘true’ concentration  , i.e. the concentration w.r.t. to the solvent is related to the ‘bulk’ concentration via,



where  is the *fluid volume fraction*, i.e. the fraction of the volume that is accessible to the fluid (and therefore to the solvents). The FEBioChem plugin will report the true concentrations of the solution.

A mixture is defined by two phases, a solid and a fluid phase. The fluid is assumed to be a solvent in which several species were dissolved. The volume fractions of the dissolved solutions are negligible and thus the solid and fluid volume fractions add up to unity.



We now consider the case that a chemical species exists as part of the solid phase. Such a species is called a *solid-bound molecule* (sbm). Since it is bound to the solid (immovable) phase, it is assumed that a solid-bound molecule does not undergo diffusion. Let us denote the apparent density of the sbm by . It then follows that the governing equation for a solid-bound molecule is given by,



where  is the mass supply for the sbm. In this context, the mass supply will be calculated from the chemical reactions.

A change in the apparent density of an SBM has an effect of the solid volume fraction, which is now defined via,



where the volume fraction of the sbm is defined via,



That is, it is the ratio of the apparent density over the true density of the sbm. The fluid volume fraction, or porosity, is then defined via .

# Using the FEBioChem plugin

Like any other FEBio plugin, the plugin must be placed in a folder and the path to the plugin must be defined in the FEBio configuration file. This file is usually called *febio.xml* and can be found in the same location as the FEBio executable. In this file, add the following line:

<import>C:\path\to\febio\plugin\FEBioChem.dll</import>

Make sure to include the full path name and file name of the plugin. When FEBio starts it will read the configuration file and load all the plugins defined therein. A message will be shown to the screen to inform the user whether the plugin was loaded successfully or not.

# Defining a FEBioChem model

For the most part an FEBioChem model is a standard FEBio model with a few modifications. These are discussed in the following sections. Note that this user’s manual assumes version 2.5 of the FEBio file format.

## Module

The *type* attribute of *Module* tag needs the value *reaction-diffusion*.

<Module type="reaction-diffusion"/>

This will inform FEBio to load the reaction-diffusion solver that is part of the FEBioChem plugin.

This module does not include convection. If you want to include the convection term, you need to use the *reaction-diffusion-convection* module.

<Module type="reaction-diffusion-convection"/>

Now, a velocity field variable is added to the model, which can be defined in the *Initial* section of the input file. See section 3.8 for more information on how to initialize the velocity field.

## Globals

The chemical species that are used in the model are defined in the *Globals* section. The species are defined identically to the way they are defined for a multiphasic problem and we refer to the FEBio user’s manual. The following caveat applies:

* A *solute* defines a chemical species that is not bound to the solid phase of a mixture. If a model doesn’t represent a mixture, each solute defines a chemical species. No child properties need to be defined for a solute.
* A *solid\_bound* defines a chemical species that is bound to the solid-phase of a mixture. Again, no child properties need to be defined for a solid bound species.

## Material

The material type needed by this plugin is *reaction-diffusion.* It needs the following material properties. Note that not all are required.

|  |  |
| --- | --- |
| **Property** | **Description** |
| *solid\_volume\_fraction* | The volume fraction of the solid phase |
| *species* | Specifies the properties of a chemical species |
| *solid\_bound\_species* | Specifies a species that remains bound to the solid phase of the mixture. |
| *reaction* | Specifies the chemical reaction. |

The *solid\_volume\_fraction* is only required when modeling a mixture. It has an initial value of zero.

### species property

Each chemical species that is used by the material must be defined in the material’s definition. (Note that this can be less than the total number of species in the model, as defined by the Globals section.). A particular species is reference by name. For each species, the *diffusivity* parameter must be defined.

<species name="A">

<diffusivity>0.05</diffusivity>

</species>

### solid\_bound\_species property

Each solid-bound chemical species that is used by the material must be included in the material’s definition. (Note that this can be less than the total number of species in the model, as defined by the Globals section.). For each solid-bound species, define the following material parameters.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Description** | **Default** |
| *rho0* | Initial apparent density (w.r.t. the mixture volume) | 0 |
| *density* | The true density of the solid-bound species | 0 |
| *rhomin* | The minimum value of the apparent density allowed | 0 |
| *rhomax* | The maximum value of the apparent density allowed | (ignored) |
| *molar\_mass* | The molar mass of the solid-bound species. | 0 |

### reaction property

The reaction property allows you to define a chemical reaction. Note that currently FEBioChem only models forward reactions. To model a reversible reaction you must specify two reactions, where the first reaction defines the forward reaction and the second defines the reverse reaction. Defining a reaction in the FEBioChem plugin is very easy since only two parameters need to be specified.

|  |  |
| --- | --- |
| **Property** | **Description** |
| *equation* | Specify the reaction equation |
| *rate\_constant* | The rate constant of the reaction rate. |

The reaction equation is defined by writing the reaction formula as follows.

* Use the name of the species as defined in the Globals section to reference it.
* A stoichiometric coefficient of 1 does not need to be defined, but otherwise it must an integer number that appears before the species’ name and separated from it via a multiplication symbol ‘\*’.
* A plus symbol ‘+’ is used to enumerate all reactants and products.
* The symbols ‘-->’ or ‘->’ can be used to separate the reactants from the products.
* The reactant or product lists can be empty.

Here are some examples. Assume that all the chemical species are properly defined in the Globals section.

A-->B

A+B-->C

2\*A+2\*B->C+B

A-->

-->B

Again, it is emphasized that all species involved in a reaction must be defined inside the material’s definition.

## Initial Values

For each species (but not solid-bound species), an initial value can be set in the *Initial* section of the FEBio input file via the *init* tag.

<init bc="c1" node\_set="set1">

<value>1.0</value>

</init>

The *bc* attribute references the particular species (or solute). In this case, *c1* references the solute with ID equal to 1. The *node\_set* references a node set that needs to be defined in the Geometry section of the input file.

## Boundary Conditions

The syntax for defining fixed or prescribed is identical for any FEBio model. Use *c[n]* to reference a particular species (or solute, again no solid-bound species). Thus, for example, *c1* references to solute with ID one, *c2* to solute with ID 2, and so forth.

An example of a fixed boundary condition looks like this.

<fix bc="c1" node\_set="FixedSet1"/>

An example of a prescribed boundary condition looks like this.

<prescribe bc="c1" node\_set="PrescribedSet1">

<scale lc="1">1.0</scale>

</prescribe>

The *node\_set* attributes references nodesets that need to be defined in the Geometry section of the input file.

## Prescribing a concentration flux

A concentration flux can be prescribed via a surface load of type *concentration flux* and is defined as follows.

<surface\_load type="concentration flux" surface="SurfaceLoad1">

<solute\_id>1</solute\_id>

<flux lc="1">-1.0</flux>

</surface\_load>

The *surface* attribute references a surface that is defined in the Geometry section. The *solute\_id* references the ID of the corresponding solute as defined in the Globals section of the input file.

## Plot Variables

The FEBioChem plugin defines several new plot variables. They are:

|  |  |
| --- | --- |
| **Plot variable** | **Description** |
| *concentration / actual concentration* | Get the true concentration of a species |
| *effective concentration* | Get the effective concentration of a species |
| *sbs concentration* | The true concentration of a solid-bound species |
| *sbs apparent density* | The apparent density of a solid-bound species |
| *solid volume fraction* | The solid volume fraction of the mixture |

Note that the variables that references a species or a solid-bound species need to specify the name of the corresponding species as part of the plot variable definition. For instance,

<var type="concentration['c1']"/>

As with any other plot variable definition in FEBio, you can define an alias.

<var type="concentration['c1']=c1/>

The alias is the name of the data field that will be stored in the plot file, and thus will show in PostView.

## Initial velocity

For reaction-diffusion-convection problems, the fluid velocity needs to be initialized. This is done in the *Initial* section of the FEBio input file.

The following example sets the x-component of the velocity vector to 1.0.

<init bc="vx" node\_set="SomeNodeSet">

<value>1.0</value>

</init>

You can also define a heterogeneous field. For instance,

<init bc="vx" node\_set="SomeNodeSet">

<value node\_data="init\_vel"/>

</init>

Here, *init\_vel* is a node data field that is defined in the *MeshData* section.