Version 2.0

User's Manual

Last Updated: April 26, 2019

Contributors
Steve Maas (steve.maas@utah.edu)
Dave Rawlins (rawlins@sci.utah.edu)
Dr. Jeffrey Weiss (jeff.weiss@utah.edu)
Dr. Gerard Ateshian (ateshian@columbia.edu)
Steven LaBelle (labellestevena@gmail.com)
Marsh Poulson (therealmarshpoulson@gmail.com)

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

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

Forum
http://mrlforums.sci.utah.edu/forums/

Development of the FEBio project is supported in part by a grant from the U.S. National Institutes of Health (R01GM083925). AngioFE is further supported by grants from the U.S. National Institutes of Health (R01HL131856, R01AR069297)

Table of Contents

1. Introduction

1.1. Overview of AngioFE

AngioFE is a growth model that represents growing neovessels as discrete line segments. The growth model is coupled to FEBio, which solves nonlinear mechanics and multiphysical problems that in turn affect the growth. AngioFE's growth model can be tailored to consider the influence of continuous variables and boundary conditions stored within FEBio.

1.1.1. Capabilities

AngioFE combines a growth model and

1.2. About this document

This document details input specification for the growth model and the finite element model.

1.3. Terminology in this document

2. Theory

2.1. Overview

This section defines the theory used to build the base growth models for AngioFE as well as relevant mechanics.

The growth model theory laid out only covers base cases. Users can interactively edit the growth model using the Grow Direction Modifiers and Generic Growth Parameters (see Chapter 4 for more detail).

2.2. Growth

AngioFE seeds angio materials with a given number of vessel fragments (1 segment long each). As new segments are added with growth, they take on a length and orientation based on a number of components expanded below.

2.2.1. Growth Velocity

New segments are added to existing segments based on user specified appearance distributions. The length of initial segments is specified by the user, while the length of additional segments is determined by the 4-parameter sigmoid curve

$$g(t) = g_0 + \frac{a}{1 + e^{-\frac{t - t_{1/2}}{b}}} \quad (4.1)$$

where g_0 is the initial total length, $t_{1/2}$ is the time required for half maximum growth along the sigmoidal curve, b is the slope of the linear portion of the curve, and a is the difference in length between the total carrying capacity and the initial total length. The user should fit experimental data using this curve. The growth curve should then be divided by $2 \times$ initial fragment # . This is to normalize the growth by the number of tips in the model i.e. for 6,000 initial fragments the growth curve (i.e. g(t)) would be divided by 12,000. AngioFE takes the growth velocity as an input. Thus, the user must determine the time derivative of the result from above i.e.

$$\frac{dg(t)}{dt} = \frac{a\left(e^{-\left(\frac{t - t_{1/2}}{b}\right)}\right)}{b\left(1 + e^{-\left(\frac{t - t_{1/2}}{b}\right)}\right)^2}$$

The user can then specify this growth using a loadcurve. An example loadcurve is provided below for the values a = 5, b = 0.5, t_{12} = 7:

```
<loadcurve id="1" type="linear">
  <point>0,8.31527336225271E-06</point>
  <point>1,6.14413685133318E-05</point>
  <point>2,0.000453958077359517</point>
  <point>3,0.00335237670756474</point>
  <point>4,0.0246650929136005</point>
  <point>5,0.176627062132911</point>
  <point>6,1.04993585403507</point>
  <point>7,2.5</point>
  <point>8,1.04993585403506</point>
  <point>9,0.176627062132911</point>
  <point>10,0.0246650929136005</point>
</loadcurve>
```

To assign this load curve, within the angio material set

```
<growth_length_over_time lc="1">1</growth_length_over_time>
```

The lc=”x” should be set to match the loadcurve id from above. The value 1 scales the loadcurve.

2.2.2. Density

Collagen density affects the length that a new segment grows. New segment lengths are scaled according to the equation:

$$v = v_0 + a_0 \exp(-a_1 c) \quad (4.2)$$

Here, c is the collagen density in mg/mL, and the values of a_0 and a_1 are set to provide an exponential decay in the growth length based on the collagen concentration. Vascular networks verified against *in-vitro* morphological measurements have used v_0 = -0.16, a_0 = 5.1605, and a_1 = 0.5112. These are the values used by AngioFE and are not currently editable at the user-level without building AngioFE from source code.

2.2.3. Direction

The growth direction for each active tip is determined by the sum

$$\boldsymbol{\psi}_n = \alpha \boldsymbol{\psi}_{n-1} + (1-\alpha)\boldsymbol{\theta} \quad (4.3)$$

Here $\boldsymbol{\psi}$ is the growth direction vector, $\boldsymbol{\theta}$ is the collagen fiber vector, and α is a scalar between 0 and 1 that weighs the contributions of the previous direction (persistence) against the local collagen fiber direction. The weight parameter is specified at the material level by the tag `<direction_weight>`. Vascular networks verified against *in-vitro* morphological measurements have used α = 0.91. These were previously done in a time step-dependent manner. The value for α should now be α = 0.36.

2.2.4. Branching

Vessels are able to branch off tip cells or established stalk cells. Within AngioFE2, the point along a growing vessel at which a branch emerges is determed as each segment is grown by the `<length_to_branch>` tag. However, the time at which the branch emerges and contributes to the stress is controlled by the `<time_to_emerge>` tag. More information on these tags is available in `

2.2.5. Anastamosis

2.3. Mechanics

2.3.1. Passive Stresses

Passive stresses arise from the resistance of the collagen matrix and vessels to deformation. The passive stress in the model is calculated as a weighted sum of the matrix and vessel stresses scaled by their volume fractions i.e.

$$\sigma_P = \phi \sigma_{Matrix} + (1-\phi)_{Vessel} \quad (4.4)$$

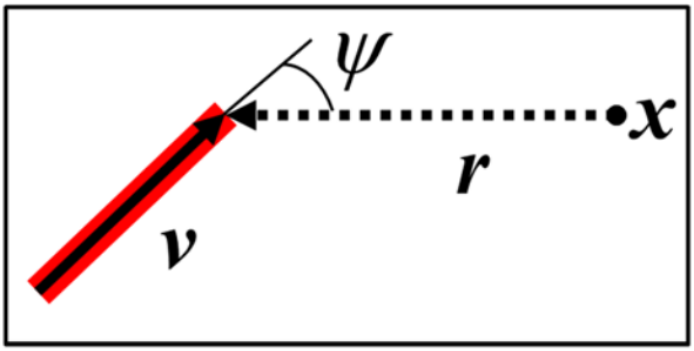
Here φ is the volume fraction of the matrix. The volume fraction of the vessel is assumed to be the complement of the matrix volume fraction.

2.3.2. Active Stresses

Body forces are applied at the tips of segments. The active stress for a tip cell is

$$\boldsymbol{\sigma}_S(\boldsymbol{x}) = \nu(\rho) a e^{-b\|\boldsymbol{r}\|} \cos^N\left(\frac{\psi}{2}\right) \hat{\boldsymbol{r}} \otimes \hat{\boldsymbol{r}} \tag{4.5}$$

Here a is the sprout stress magnitude, N is an exponential used to tighten the fall off of the exponential term. The b terms controls the range of the exponential falloff. The stress is scaled by the density using $\nu(\rho)$ from equation (4.2). The stress calculation at a point $\boldsymbol{x}$ from a single tip is presented below. The tip is indicated in red. The distance between $\boldsymbol{x}$ and the tip cell is given by the vector $\boldsymbol{r}$. The angle between the axis of the tip cell and the distance vector is given by ψ.



Previously used values for these parameters are available below:

Measure	Value	Units
Sprout stress (a)	0.025	kg/μm x day^2
Cosine exponential (N)	2	
Sprout stress range (b)	250	μm

2.4. Units

The units need to be consistent with the values used for the materials from FEBio.(unless you are willing to multiply by the conversion factor whenever the values are used) In general the units used are:

Measure	Units
Length	Micron
Time	Days
Mass	Kilogram

The multipliers when converting units are:

From	To	Multiplier
N (kg x meter / sec^2)	(kg x micron)/day^2	7.46496 x 10^{15}
Meters/sec^2	Microns/day^2	7.46496 x 10^{15}
Pascal	kg/(micron x day^2)	7.46496 x 10^3
OsmoticCoefficient		1

Some constants are:

Constant	Value	Units
Acceleration	7.323 x 10^{15}	microns/day^2
R	6.206 x 10^{22}	kilogram x $micron^2$ / (day^2 x Kelvin x mol)
Faraday Constant	1.116	Amp x day / mol

3. Running FEBio+AngioFE

3.1. Configuring FEBio to find the Plugin

A configuration file which allows FEBio to find the AngioFE is required. An example of this file febio.xml is:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<febio_config version="1.0">
      <linear_solver type="pardiso"></linear_solver>
      <import>C:\Users\mp4\Documents\AngioFE2\VS2010\Debug\AngioFE2.dll
</import>
</febio_config>
```

Make sure the linear solver is supported by your build of FEBio. Make sure the import line points to the AngioFE2 shared object. Additionally, each build of the plugin can only be run with a build of FEBio that is built with the same complier for the same operating system.

3.2. Running a Simulation with the Plugin

When run from the commandline the command should be like:
(Windows) Run febio.exe –i control.feb –cnf febio.xml task=angio angiofe.txt
(Linux) ./ febio.exe –i control.feb –cnf febio.xml task=angio angiofe.txt
Angiofe.txt should be in the same directory as control.feb the contents of this file will be ignored. (this file may be empty) Angiofe.txt may be removed in a future version of the plugin.

4. Material – Mechanics and Initialization

The current version of this plugin expects the .feb file to be in the FEBio 2.5 spec format.

4.1. Passive Stress Modifiers

The vessels and matrix are modeled by elastic materials. If you don't want to model these materials set.

```
<composite_material>0</composite_material>
```

Otherwise make sure

```
<composite_material>1</composite_material>
```

is set.

Note that these may be modeled with viscoelastic materials which may depend on the results of previous stress calculations; if you want to simulate the same vascular network make sure these material parameters are set to accommodate the new step size.

4.1.1. Matrix Elastic Material

This material models the effects of the matrix on the rest of the simulation. This must be a viscoelastic material. All of the inner parameters are from the underlying FEBio materials see FEBio User Manual Section 4.3.

e.g.

```
<matrix type="viscoelastic">
      <t1>1.25e-5</t1>
      <g0>0.0</g0>
      <g1>1.0</g1>
      <elastic type="EFD neo-Hookean">
            <E>2.58e5</E>
            <v>0.</v>
            <beta>2.5,2.5,2.5</beta>
            <ksi>2.58e6, 2.58e6, 2.58e6</ksi>
      </elastic>
</matrix>
```

4.1.2. Vessel Elastic Material

This material models the effects the vessels themselves have on the finite element properties of the matrix. This must be a viscoelastic material. All of the inner parameters are from the underlying FEBio materials see FEBio User Manual Section 4.3.

e.g.

```
<vessel type="viscoelastic">
      <t1>1.25e-5</t1>
      <g0>0.0</g0>
      <g1>1.0</g1>
      <elastic type="neo-Hookean">
            <E>2.58e7</E>
            <v>0.0</v>
      </elastic>
</vessel>
```

4.1.3. Vessel Width

This parameter affects the weighting for the vessel and matrix submaterials. This parameter is in um.
e.g.

```
<vessel_width>7.0</vessel_width>
```

4.1.4. ECM Density

The density effects the distance that segments grow; ECM's that are denser will impede growth whereas less dense matrices will have more segment growth. This is calculated on a per node basis so this can vary within a given material as contraction of the gel occurs. This density is not the same as the density of the material and is only used in modifying sprout stress and velocity.
e.g.

```
<matrix_density>3</matrix_density>
```

4.1.5. ECM Seeder

The ecm seeder determines how the densities and anisotropies are initialized within the materials. This mostly effects how the densities and anisotropies are initialized on the boundary between materials. There are 3 valid values for this parameter. The default for this parameter is 0. This should work for most single material simulations. 0, constant mode, all nodes in material are set to the specified values. 1, specified mode, read in values at integration points and then interpolate them to the nodes. 2, no overwrite mode, if the densities and anisotropies have not been previously set, set them using this material's values.
e.g.

```
<ecm_seeder>2</ecm_seeder>
```

4.2. Active Stresses

4.2.1. Sprout Parameters

A is the magnitude of the sprout this is used in applying stresses created by the sprouts. B is the range of the sprout this effects the falloff of the other parameters. N is the sprout width this also effects the amount of stress applied to each sprout.
e.g.

```
<a>0.025</a>
<b>0.004</b>
<N>2.0</N>
```

Experimentally we have observed that the contraction (and thus stress) of collagen gels does not accrue until after a few days of growth. Thus, it is likely that you will want to set the sprout stress magnitude, <a>, to a load curve to capture this time-dependent behavior.

4.3. Intra-Material Boundary Conditions

The boundary conditions discussed in this section are not the same as the mechanical boundary conditions. The boundary conditions discussed here are the boundaries between materials. (two materials share some nodes)

4.3.1. Non Angio Boundaries

This is the boundary between an angio material and a non-angio material, this controls what segments do when they encounter this boundary. The options of what the segments do currently is: stop or bounce. When a segment bounces it will be reflected about the normal of the face that it finds.

e.g.

```
<boundary_condition type="bouncy">
<boundary_condition type="stop">
```

4.3.2. Angio-Angio Boundaries

Angio material to angio material boundaries have two options: either be handled the same as non-angio boundaries, or allow vessels to pass through this boundary. Segments that pass through the boundary will acquire the growth characteristics of the material that they have grown into. Angio boundary groups are an additional check on whether a segment can grow into another angio material. Angio boundary groups is interpreted as a binary number if two group numbers share any bits a segment can grow between these materials. Otherwise segments cannot grow between the angio materials and the non-angio boundary condition will be used. (This can be used to help force a segment to be evaluated in a very thin material)

e.g.

```
<boundary_condition type="bouncy">
      <angio_boundary_groups>1</angio_boundary_groups>
      <mbc type="pass_through"/>
</boundary_condition>
```

Or

```
<boundary_condition type="bouncy">
      <angio_boundary_groups>1</angio_boundary_groups>
      <mbc type="same"/>
</boundary_condition>
```

4.4. Fragment Seeders

Fragment Seeders determine the position of the initial vessels fragments within a material.

4.4.1. MultiDomain Seeder

This fragment seeder distributes the initial vessel fragments between elements of all angio materials. When each fragment is placed each element has an equal probability of containing this vessel regardless of element size. This has two parameters: number of fragments, and initial vessel length.

e.g.

```
<fragment_seeder type="MD">
      <number_fragments>25</number_fragments>
      <initial_vessel_length>42.0</initial_vessel_length>
</fragment_seeder>
```

4.4.2. Volumetric Seeder

This fragment seeder distributes the initial vessel fragments between all elements. When each fragment is placed each element has a probability proportional to the volume of the element of containing this vessel with respect to the volume of the material. This has two parameters: number of fragments, and initial vessel length.

e.g.

```
<fragment_seeder type="MDbyVolume">
      <number_fragments>25</number_fragments>
      <initial_vessel_length>42.0</initial_vessel_length>
</fragment_seeder>
```

4.5. Vascular Stress Modifiers

4.5.1. Vascular Stress Radius

This is a parameter that allows you to control the granularity of how large an area each point should look to find active tips that may effect the stress of the current point. The units are the units of distance. Computation time will increase with radius.

e.g.

```
<stress_radius>2500</stress_radius>
```

4.6. Fibril Alignment

The fibril alignment is how the fibers are oriented within a given material. The fibers are modeled by adjusting a material axes. These values are stored at the integeration points of the material. During initialization fibers may be interpolated from the nodes to the integration points.

4.6.1. Random Alignment

Random alignment means that the fiber directions will be randomly generated before the first grow step occurs. All directions have an equal probability of being chosen. This mode interpolates the randomly generated fiber orientations from the nodes to the integration points. This is the previous mode but with interpolation. This mode has the highest error as the interpolation from the nodes to the integration points is not lossless (large angles between fibers may change during interpolation).

e.g.

```
<fiber_initializer type="random_fiber_initializer"/>
```

4.6.2. Random Alignment Non-Mangling

This mode transforms the fiber orientations at each integration point by rotating them to a random orientation. This is the recommended mode for simulating randomly aligned fibers.

e.g.

```
<fiber_initializer type="random_fiber_initializer_non_mangling"/>
```

4.6.3. Random Alignment per Element

This mode generates the fiber orientations on a per element basis. This means the orientations within each element are rotated by the same amount. This means that there will be a pocket within each element that has very similar orientations.
e.g.

```
<fiber_initializer type="random_fiber_initializer_pe"/>
```

4.6.4. No Change to Alignment

This doesn't change the material axes supplied by febio. This allows for simulating a case where the fibers are aligned.
e.g
within angio material:

```
<mat_axis type="vector" >
      <a>1,0,0</a>
      <d>0,0.5,0</d>
</mat_axis>
…
<fiber_initializer type="null_fiber_initializer"/>
```

5. Material – Growth Attributes

This section covers attributes set at the material level that allow the user to define sophisticated growth networks and set growth parameters as functions of state variables and distributions.

5.1. Growth Modifier

The growth modifiers are the parameters that effect the distance that segments grow within an angio material.

5.1.1. Growth Length over Time

This is the length that a segment would grow in a 3mg/ml density gel over 1 unit of time. Previously this was calculated using a formula from Edgar. This can be set from a loadcurve to approximate this function. This allows the user to control the granularity of the approximation rather than letting the timestep implicitly determine the approximation.
e.g.

```
<growth_length_over_time lc="1">1</growth_length_over_time>
```

5.1.2. Length Adjustment

This parameter that is a global multiplier for segment length.
e.g.

```
<length_adjustment>1.0</length_adjustment>
```

5.2. Probability Distributions

The probability distributions are used as parameters for various parts of the branching model. The parameters of these distributions can be changed at any given mechanical point safely. The distributions will reroll any values that are produced that are below zero. (branches cannot happen in the past, and negative lengths are not allowed)

5.2.1. Normal Distribution

The Normal Distribution is one of the provided distributions. This distribution has 2 parameters: mean, and stddev. The mean is the mean of the distribution. The stddev is the standard deviation of the distribution.

Wikipedia Page C++ Documentation

e.g.

```
<length_to_branch type="normal_distribution">
      <mean>200</mean>
      <stddev>5</stddev>
</length_to_branch>
```

5.2.2. Uniform Distribution

The Uniform Distribution is one of the provided distributions. This distribution has 3 parameters: a, b, and time_clamped. If time_clamped is false a and b are the ends of the distribution. Otherwise the ends of the distribution are: a, and b-time. (1 is for time_clamped true, 0 is for time_clamped false)

Wikipedia Page C++ Documentation

e.g.

```
<time_to_emerge type="uniform_distribution">
      <a>0</a>
      <b>10</b>
      <time_clamped>1</time_clamped>
</time_to_emerge>
```

5.2.3. Exponential Distribution

The Exponential Distribution is another provided distrution. It has 2 parameters: mult and lambda. Lambda is the rate of this distribution. Mult is a scale parameter that scales the result of the distribution.

Wikipedia Page C++ Documentation

e.g.

```
<time_to_emerge type="exponential_distribution">
      <lambda>0.5</lambda>
      <mult>1</mult>
</time_to_emerge>
```

5.2.4. Cauchy Distribution

The Cauchy Distribution is another provided distribution. It has 2 parameters: a, and b. A is location. B is scale which must be greater than 0.

Wikipedia Page C++ Documentation
e.g.
```
<time_to_emerge type="cauchy_distribution">
      <a>0.5</a>
      <b>0.5</b>
</time_to_emerge>
```

5.2.5. Chi Squared Distribution

The Chi Squared Distribution is another provided distrution. It has 2 parameters: dof, and mult. dof is degrees of freedom. Mult is scale.
Wikipedia Page C++ Documentation
e.g.
```
<time_to_emerge type="chi_squared_distribution">
      <dof>3.0</dof>
      <mult>0.5</mult>
</time_to_emerge>
```

5.2.6. Weibull Distribution

The Weibull Distribution is another provided distribution. It has 2 parameters: a, and b. A is shape. B is scale.
Wikipedia Page C++ Documentation
e.g.
```
<time_to_emerge type="weibull_distribution">
      <a>2.0</a>
      <b>0.5</b>
</time_to_emerge>
```

5.2.7. Gamma Distribution

The Gamma Distribution is another provided distribution. It has 2 parameters: alpha, and beta. Alpha is shape. Beta is rate.
Wikipedia Page C++ Documentation
e.g.
```
<time_to_emerge type="gamma_distribution">
      <alpha>2.0</alpha>
      <beta>0.5</beta>
</time_to_emerge>
```

5.3. Grow Direction Modifiers(GDM)

Grow direction modifiers modify the direction that a segment would grow. These modifiers are specified as: `<gdm type="default_grow_direction"/>` tags within the `<grow_direction_modifiers>` tags within the material. At least one grow direction modifier must be specified; also not all grow direction modifiers must be used. The order these are in the material will specify the order that these modifiers are applied to a vessel tip. Grow Direction Modifiers may have any number of bind points (including zero). These bind points use a tree of Generic Growth Parameter to modify the inputs to the current GDM (This tree can be empty).

Each section will specify the type of each parameter; for more information see the section on Generic Growth Parameters.
e.g.

```
<grow_direction_modifiers>
      <gdm type="default_grow_direction"></gdm>
      <gdm type="anastamosis_grow_direction"></gdm>
      <gdm type="branch_grow_direction"></gdm>
</grow_direction_modifiers>
```

5.3.1. Default Grow Direction

Default grow direction calculates the current grow direction based on the previous direction, and the collagen direction. The mixture between the two is calculated by the weight interpolation times the current timestep. This or Selecting Grow Direction should be the first grow direction modifier in the material (not both however). This GDM has three bind points: collagen_direction(vec3d), previous_direction(vec3d), and weight_interpolation(double).
e.g. `<gdm type="default_grow_direction"/>`

5.3.1.1. Default Grow Direction Methods:

Mix3d: Choose the method to mix the previous and collagen direction are combined to determine the new direction.

Scaled Vector Components: This is the legacy method. A linear interpolation between the vectors determines the new collagen direction e.g.

$$\boldsymbol{\psi}_{n+1} = \alpha\boldsymbol{\theta}_n + (1-\alpha)\boldsymbol{\psi}_n \tag{4.6}$$

The resulting vector does not rotate linearly with the weight so this is no longer the default method.
e.g.

```
<gdm type=”default_grow_direction”>
      <mix_3d>0</mix_3d>
</gdm>
```

Scaled Vector Rotation: This is the default method and will be called if the mix_3d option is omitted. The scale in this method scales a rotation normal to the collagen and previous directions but within the plane that spans them. Thus, the scale linearly determines the relative rotation from the previous direction to the current direction. First the normal to the plane by the vectors is determined

$$\boldsymbol{n} = \boldsymbol{\psi} \times \boldsymbol{\theta} \tag{4.7}$$

The angle φ between the previous direction and influential direction is determined from the dot product then a rotation about the normal vector by the angle φ scaled by α:

$$\boldsymbol{\psi}_{n+1} = \boldsymbol{R}(\alpha\varphi, \boldsymbol{n})\boldsymbol{\psi}_n \tag{4.8}$$

This method guarantees that the relative change in direction is linear with respect to the scaling weight.
e.g.

```
<gdm type=”default_grow_direction”>
```

```
      <mix_3d>1</mix_3d>
</gdm>
```

5.3.2. Selecting Grow Direction

Same as default grow direction but prescribes growth along the axis of the collagen direction rather than along the collagen direction. If the angle between the collagen direction and the previous direction is large then the sign of the direction will be flipped. This GDM has three bind points: collagen_direction(vec3), previous_direction(vec3d), and weight_interpolation(double).
e.g.

```
<gdm type="selecting_grow_direction"/>
```

5.3.2.1. Selecting Grow Direction Methods

Mix3d: Choose the method to mix the previous and collagen direction are combined to determine the new direction.

Scaled Vector Components: This is the legacy method. A linear interpolation between the vectors determines the new collagen direction e.g.

$$\psi_{new} = \psi(1-\alpha) + \alpha\theta$$

The resulting vector does not rotate linearly with the weight so this is no longer the default method.
e.g.

```
<gdm type=”default_grow_direction”>
      <mix_3d>0</mix_3d>
</gdm>
```

Scaled Vector Rotation: This is the default method and will be called if the mix_3d option is omitted. The scale in this method scales a rotation normal to the collagen and previous directions but within the plane that spans them. Thus, the scale linearly determines the relative rotation from the previous direction to the current direction
e.g.

```
<gdm type=”default_grow_direction”>
      <mix_3d>1</mix_3d>
</gdm>
```

5.3.3. Branching Grow Direction

Branch grow direction changes the direction of a given tip if that tip is a branch. The main purpose of this grow direction modifier is in what order this is performed in relation to the other grow direction modifiers. This GDM has two bind points: collagen_direction(vec3d), and previous_direction(vec3d).
e.g. `<gdm type="branch_grow_direction"/>`

5.3.4. Random Branching Grow Direction

Branch grow direction changes the direction of a given tip if that tip is a branch. This makes the direction the branch grows in completely random.
e.g. `<gdm type="random_branch_grow_direction"/>`

5.3.5. Random Theta Branching Grow Direction

Branch grow direction changes the direction of a given tip if that tip is a branch. This makes the direction the branch grows in the same as branch grow direction but the result is spun about the direction of the parent segment.

e.g. `<gdm type="random_theta_branch_grow_direction"/>`

5.3.6. Anastamosis

The anastomosis grow direction modifier simulates the change in direction due to two vessels being within a given distance of each other. This modifier has two parameters: search_radius, and search multiplier. Search_radius is in the units used by the simulation. Search multiplier controls the search radius in proportion to the growth length of the current segment. With a value of one the total search radius is the search_radius plus the grow distance of the current segment. Zero is another reasonable value for this parameter. Search_multiplier defaults to 1. Anastamosis does not effect the first segment in a branch. Additionally, segments won't fuse with vessels that originate from the same fragment.

e.g.

```
<gdm type="anastamosis_grow_direction">
      <search_radius>100.0</search_radius>
      <search_multiplier>1.0</ search_multiplier >
</gdm>
```

5.3.7. Density Gradient

The density is stored at the nodes by this plugin; there are parameters that allow you to specify which material contributes to the density value at the nodes along an angio-angio material boundary. For this modifier if a tip is in a region with a density gradient above the threshold the tip will grow perpendicular to the direction of the density gradient. If the density gradient is below the threshold this modifier will do nothing. This GDM has one bind point: threshold_scale(double) Units are in $[M/L^4]$.

e.g.

```
<gdm type="gradient_grow_direction">
      <threshold>0.005</threshold>
</gdm>
```

5.3.8. Unit Length

Sets the scale for the unit length since the FEBio model units are never explicitly stated. This scales the length unit of growth parameters to match lengths in the model. Thus, for a model in micrometers with vessel data in micrometers, the unit length is 1. If the FE model were in mm and the vessel data were all in um, then the scale would be set to 1/1000. This must be used in all materials where growth is wanted.

e.g.

```
<gdm type="unit_length">
      <length_modifier>1e-3</length_modifier>
</gdm>
```

5.3.9. Density Length

Modfies the length of the segment by the current density of the tip location. This GDM has one bind point: density_scale(double).
e.g.

```
<gdm type="density_length">
</gdm>
```

5.3.10. Ref Density Length

Modifies the length of the segment by the densitry of the tip location in the reference configuration. This GDM has one bind point: ref_density_scale(double). This should be used when large deformations are imposed on the angio material boundaries that result in high apparent densities.
e.g.

```
<gdm type="ref_density_length">
</gdm>
```

5.3.11. Segment Length

This modifies that length of the segment scaled by the growth length over time and the size of the current angio timestep.
e.g.

```
<gdm type="segment_length">
</gdm>
```

5.3.12. Base Fiber Aware Grow Direction Modifier

5.4. Generic Growth Parameter(GGP)

These allow the user to write small programs that modify the values at specific bind points in various GDM's. The bind points are somewhat similar to opengl shader programs and some of the language that is used is borrowed from opengl shaders. A tree of these will allow the user to use various data at the nodes in the simulation to modify various growth parameters. The data these interface with can be from: AngioFE, FEBio, or another plugin(as long as it properly registers whatever data is needed). An example of an application of this is allowing segments to grow through a density gradient in the presence of a particular concentration of VEGF. In general all GGP's have one optional parameter child that is another GGP(there may be other parameters). Additionally the child parameter will be the final sub GGP that is called at the current tree level. The following example uses the threshold_scale bind point in the gradient grow direction GDM.
e.g.

```
<gdm type="gradient_grow_direction">
      <threshold_scale type="plot2_ggp">
            <field_name>matrix weight</field_name>
            <child type="threshold_ggp">
                  <threshold type="setter_ggp">
                        <m11>1.0</m11>
                        <m12>0.0</m12>
                        <m13>0.0</m13>
                        <m21>0.0</m21>
                        <m22>1.0</m22>
```

```
                        <m23>0.0</m23>
                        <m31>0.0</m31>
                        <m32>0.0</m32>
                        <m33>1.0</m33>

                        <in_x>1.0</in_x>
                        <in_y>0.0</in_y>
                        <in_z>0.0</in_z>
                  </threshold>
                  <statement type="cos_ggp"></statement>
            </child>
      </threshold_scale>
</gdm>
```

5.4.1. Matrix Converter GGP

This class allows you to swizzle the components of a matrix or do matrix multiplication. It does this by unrolling the input matrix into a column vector [11,12,13,21,..33]. This column vector gets multiplied by a 9x9 matrix that is set in this material. The components of this matrix are named m11, m12,… m99. This matrix starts out as an identity matrix, only the componets that need to be changed need to be set.

e.g.

```
<gdm type="default_grow_direction">
      <collagen_direction type="forked_ggp">
            <child type="matrix_converter_ggp">
                  <m51>1.0</m51>
                  <m55>0.0</m55>
                  <m91>1.0</m91>
                  <m99>0.0</m99>
            </child>
            <nest type="plot2_ggp">
                  <field_name>matrix weight</field_name>
            </nest>
      </collagen_direction>
</gdm>
```

5.4.2. Forked GGP

This GGP allows a subtree to be executed before this recently computed result is multiplied by child. The subtree is named nest and is a required parameter.

e.g.

```
<gdm type="default_grow_direction">
      <collagen_direction type="forked_ggp">
            <child type="matrix_converter_ggp">
                  <m51>1.0</m51>
                  <m55>0.0</m55>
                  <m91>1.0</m91>
                  <m99>0.0</m99>
            </child>
            <nest type="plot2_ggp">
                  <field_name>matrix weight</field_name>
```

```
            </nest>
      </collagen_direction>
</gdm>
```

5.4.3. Eigen Values GGP

Computes the eigen values and stores them in the first column of the matrix. The input to the matrix must be a symmetric matrix or the output of this GGP is undefined. There is no correspondence between the order of the eigen values and the eigen vectors.
e.g.

```
<child type="eigen_values_ggp"/>
```

5.4.4. Eigen Vectors GGP

Computes the eigen vectors and stores them in the collumns of the matrix. The input to the matrix must be a symmetric matrix or the output of this GGP is undefined.
e.g.

```
<child type="eigen_vectors_ggp"/>
```

5.4.5. Cross GGP

Computes the cross product of its two required parameters: v1, and v2. The vectors are read out of the diagonals of the input matrices. The resulting vector is returned in the diagonal of the return matrix.(all other parameters are zeroed)
e.g.

```
<child type="cross_ggp">
      <v1 type="plot2_ggp">
            <field_name>matrix weight</field_name>
      </v1>
      <v2 type="plot2_ggp">
            <field_name>matrix weight</field_name>
      </v2>
</child>
```

5.4.6. Threshold GGP

This GGP allows for conditional inclusion of a matrix. This has two required parameters: threshold, and statement. If the x component of the result of threshold times (1,0,0) is greater than zero then statement will be executed and multiplied by the input before it is used as an input to child.

e.g.

```
<gdm type="gradient_grow_direction">
      <threshold_scale type="plot2_ggp">
            <field_name>matrix weight</field_name>
            <child type="threshold_ggp">
                  <threshold type="setter_ggp">
                        <m11>1.0</m11>
```

```
                        <m12>0.0</m12>
                        <m13>0.0</m13>
                        <m21>0.0</m21>
                        <m22>1.0</m22>
                        <m23>0.0</m23>
                        <m31>0.0</m31>
                        <m32>0.0</m32>
                        <m33>1.0</m33>

                        <in_x>1.0</in_x>
                        <in_y>0.0</in_y>
                        <in_z>0.0</in_z>
                    </threshold>
                    <statement type="cos_ggp">
                    </statement>
                </child>
            </threshold_scale>
</gdm>
```

5.4.7. ArcCos GGP

Takes the arccosine of each element of the matrix.
e.g.

```
<statement type="arccos_ggp">
</statement>
```

5.4.8. ArcSin GGP

Takes the arcsine of each element of the matrix.
e.g.

```
<statement type="arcsin_ggp">
</statement>
```

5.4.9. Cos GGP

Takes the cosine of each element of the matrix.
e.g.

```
<statement type="cos_ggp">
</statement>
```

5.4.10. Sin GGP

Takes the sine of each element of the matrix.
e.g.

```
<statement type="sin_ggp">
</statement>
```

5.4.11. Matrix Inverse GGP

Computes the inverse matrix of the input matrix and returns this. If the matrix is not invertable this operation is not defined.

```
<statement type="matrix_inverse_ggp">
</statement>
```

5.4.12. UnitDiagonalGGP

Turns the diagonal of this GGP's input matrix into a unit vector.
e.g.

```
<child type="unit_diagonal_ggp"/>
```

5.4.13. DirectionChangeGGP

This reverse the direction of a segment if the input vector is $\frac{\pi}{2}$ or more away from vector. Used to force growth to be one way across a gradient.
e.g.

```
<child type="direction_change_ggp">
      <vector type="gradient_plot2_ggp">
            <field_name>solute 1 concentration</field_name>
      </vector>
</child>
```

5.5. Branching Generators

Branching Generators determine how/when fragments branch within the plugin.

5.5.1. Pseudo Deferred Fragment Brancher

The pseudo deferred fragment brancher allows the same vascular network to be approximated as long as the timestep is the only thing that differs between two simulations. This brancher has two parameters, both are probability distributions, length to branch, this is the length that a vessel will grow before a branch is created. The other parameter is time to emerge this specifies how long a branch point will wait after the length requirement has been met before a branch is created
e.g.

```
<brancher type="psuedo_defered_branch">
      <length_to_branch type="normal_distribution">
            <mean>200</mean>
            <stddev>5</stddev>
      </length_to_branch>
      <time_to_emerge type="normal_distribution">
            <mean>0.5</mean>
            <stddev>0.125</stddev>
      </time_to_emerge>
</brancher>
```

5.6. Advanced GGP's

The interface to these GGP's may change without warning. These are mostly used in testing the project and should not be needed by normal finite element simulations.

5.6.1. Plot2GGP

Must be used with the plotfile2 output format. This allows the import of any parameter that is currently being output. A double will be read in to matrix position 1,1 of the matrix. A vector will be read in to the diagonal of the matrix. All matrices will be expanded to a 3x3 matrix. All other values in the matrix will be left alone.

e.g.

```
<weight_interpolation type="plot2_ggp">
      <field_name>matrix weight</field_name>
</weight_interpolation>
```

5.6.2. GradientPlot2GGP

Must be used with the plotfile2 output format. Must be used with a signel valued variable(no vector or matrix types) Calculates the gradient of the field and stores it in the diagonal of the matrix.

```
<vector type="gradient_plot2_ggp">
                  <field_name>solute 1 concentration</field_name>
</vector>
```

5.6.3. NodalDataGGP

This is another way to access data from FEBio, for all nodes in the mesh the accessed value must be defined. This has 2 parameters: field_name, and offset. Field_name is which field can be read out of. Offset is the number of DOFs to go from the base value from field name. This GGP reads a single value in to position 1,1 of the matrix, all other values in the matrix will remain the same.

e.g.

```
<density_scale type="nodal_data_ggp">
      <field_name>displacement</field_name>
      <offset>1</offset>
</density_scale>
```

5.6.4. SetterGGP

This GPP sets the input vector and matrix.

e.g.

```
<threshold type="setter_ggp">
      <m11>1.0</m11>
      <m12>0.0</m12>
      <m13>0.0</m13>
      <m21>0.0</m21>
      <m22>1.0</m22>
      <m23>0.0</m23>
      <m31>0.0</m31>
      <m32>0.0</m32>
      <m33>1.0</m33>
```

```
      <in_x>1.0</in_x>
      <in_y>0.0</in_y>
      <in_z>0.0</in_z>
</threshold>
```

5.6.5. MatrixSetterGGP

Sets only the matrix.
e.g.

```
<statement type="matrix_setter_ggp">
      <m11>1.0</m11>
      <m12>0.0</m12>
      <m13>0.0</m13>
      <m21>0.0</m21>
      <m22>1.0</m22>
      <m23>0.0</m23>
      <m31>0.0</m31>
      <m32>0.0</m32>
      <m33>1.0</m33>
</statement>
```

5.6.6. AssertGGP

This only works with a plugin built with debug mode. Verifies that the values of the matrix are within the tolerance.
e.g.

```
<child type="assert_ggp">
      <m11>0.66666</m11>
      <m12>0.33333</m12>
      <m13>0.66666</m13>
      <m21>-0.4472135</m21>
      <m22>0.894427</m22>
      <m23>0.0</m23>
      <m31>-0.59628</m31>
      <m32>-0.29814</m32>
      <m33>0.7435</m33>
      <tolerance>0.2</tolerance>
</child>
```

6. Global Constants

The global constants for AngioFE go in the `<Globals> <Constants>` section in the .feb file.

6.1. Seed

The seed is the seed for the underlying random engine. If this parameter is changed the vascular network will change even if all other parameters remain the same. This parameter will be interpreted as an integer. When simulating multiple models with different seeds it is suggested that the difference in seeds be on the same order of magnitude as the number of elements are within the angio material
e.g. `<seed>1393430476</seed>`

6.2. Toggle IO

This parameter disables all of the custom files that are created by this plugin. Set this parameter to a nonzero value. If this parameter is not specified all of the files will be created.
e.g. `<no_io>1</no_io>`

6.3. Independent Angio Growth Steps

This parameter specified gives the id of a loadcurve in the LoadData section in the .feb file. If this parameter is not specified, the plugin will do a growth step after each mechanical step. The points in the loadcurve will be used as must points for when the growth steps will be performed.
e.g. to assign loadcurve 1, `<angio_time_step_curve>1</angio_time_step_curve>`

7. Output Data

7.1. Files

This section describes the possible outputs of this plugin. The two main modes of output are creating files in the same directory the run was started from, or adding additional data to the .xplt file which can be viewed with heat maps.

7.1.1. Log

Statistics from this plugin are recorded in out_log.csv this file can be opened by Excel. The values recorded are: Time, Material, Segments, Total Length, Vessels, Branch Points, Aamosis, Active Tips, and Sprouts. Time is the time of this data. Material is the material id -1. Total Length is the total vessel length within this material. Vessels is the number of vessels within this material, this increases as branches are created and decreases as vessels fuse together due to anastomosis. Branch Points is the number of branches that have happened at the current time. Anastamosis is the number of tips that have fused to another vessel. Active tips are the number of tips within the current material that will grow in the next grow step.

7.1.2. Vessel State

The file out_vessel_state.ang2 contains a record of the vascular network over time. To import this file in PostView click tools>Import lines and select the file. (make sure that the tools tab is enabled in PostView)

7.1.3. Final Vessel State

Contains data for each tip at the final time point recorded in final_vessels.csv. This file contains data from the tips such as position. This will be expanded in the future to contain tip variables such as concentrations. It can be opened by the polar_plot.py python script to generate orientation distribution functions for segment growth.

7.1.4. XPLT

Some output options can be specified to show up in the heat maps in the .xplt file. These can be specified in the `<Output><plotfile type="febio">` section within the .feb file.

`<var type="angio stress"/>`
This tag will output the stress that the vascular network creates.

`<var type="vessel stress"/>`
This tag will output the stress that is from the vessel submaterial.

`<var type="matrix stress"/>`
This tag will output the stress that is from the matrix submaterial.

`<var type="vessel weight"/>`
This tag will output the weighting of the vessel material at each point.

`<var type="matrix weight"/>`
This tag will output the weighting of the matrix material at each point.

`<var type="matrix visco stress"/>`
This tag will output the matrix submaterial's viscoelastic submaterial's stress.

`<var type="matrix elastic stress"/>`
This tag will output the matrix submaterial's elastic submaterial stress.

`<var type="angio ECM alpha"/>`
This tag will output the alpha at each node. This is a measure of the volume of vessels at each node in the matrix.

Note the fibers used in the simulation match the fibers used in FEBio so the tag to output them is from febio and is
`<var type="fiber vector "/>`

e.g.

```
<Output>
     <plotfile type="febio">
          <var type="displacement"/>
          <var type="stress"/>
          <var type="angio stress"/>
          <var type="vessel stress"/>
          <var type="matrix stress"/>
          <var type="vessel weight"/>
          <var type="matrix weight"/>
          <var type="matrix tangent"/>
          <var type="matrix visco stress"/>
          <var type="matrix elastic stress"/>
          <var type="fiber vector"/>
          <var type="angio collagen fiber"/>
```

```
            <var type="angio ECM alpha"/>
        </plotfile>
</Output>
```

8. Advanced Features

The advanced features should not need to normally be used but they may be needed for advanced use cases. These can change at any time; very little consideration will be given to changing this between releases.

8.1. Active Tip Threshold

This controls the switch from the brute force method to using the kd-tree to find the segments within a radius for stress calculations. This defaults to 500 which from testing is close to the amount of segments needed to make the compuations done with the kd-tree faster.
e.g.

```
<active_tip_threshold>350</act_tip_threshold>
```

8.2. Vessel File Format 2

This format is the output format for the vessel file that is output by this plugin. The file has extension .ang2. All of the coordinates that are stored in this file are relative to the reference configuration. All numbers in this file are written in a little endian format. All integers in this file are unsigned. All floating point numbers are in the IEEE 754 format. This file starts with the magic number 0xfdb97531. Next in the file is the version which is 4 bytes, the current version is 0. The remained of the file is organized into sections with one section per mechanical timestep. Each section starts with the number of segments created in this timestep, this is stored as a 4 byte integer. Next in the section is a floating point number is the start time of the current mechanical timestep. Next in the file is the collection of segments that grew this timestep. Each segment is represented with 6 floating point numbers, x_0, y_0, z_0, x_1, y_1, z_1.

Appendix A. All Together

An example of a material

```
<Material>
      <material id="1" name="Material01" type="angio">
            <a>0.025</a>
            <b>0.004</b>
            <N>2.0</N>
            <matrix_density>3</matrix_density>
            <composite_material>1</composite_material>
            <length_adjustment>1.0</length_adjustment>
            <matrix type="viscoelastic">
                  <t1>1.25e-5</t1>
                  <g0>0.0</g0>
                  <g1>1.0</g1>
                  <elastic type="EFD neo-Hookean">
                        <E>2.58e5</E>
                        <v>0.0</v>
                        <beta>2.5,2.5,2.5</beta>
                        <ksi>2.58e6, 2.58e6, 2.58e6 </ksi>
                  </elastic>
            </matrix>
            <common_properties type="angio_properties">
                  <fiber_initializer type="null_fiber_initializer"/>
                  <fragment_seeder type="MD">
                        <number_fragments>25</number_fragments>

      <initial_vessel_length>42.0</initial_vessel_length>
                  </fragment_seeder>

                  <vessel type="viscoelastic">
                        <t1>1.25e-5</t1>
                        <g0>0.0</g0>
                        <g1>1.0</g1>
                        <elastic type="neo-Hookean">
                              <E>2.58e7</E>
                              <v>0.0</v>
                        </elastic>
                  </vessel>

                  <brancher type="psuedo_defered_branch">
                        <length_to_branch type="normal_distribution">
                              <mean>200</mean>
                              <stddev>5</stddev>
                        </length_to_branch>
                        <time_to_emerge type="normal_distribution">
                              <mean>0.5</mean>
                              <stddev>0.125</stddev>
                        </time_to_emerge>
                  </brancher>
                  <grow_direction_modifiers>
```

```
                    <gdm type="unit_length">
                    </gdm>
                    <gdm type="density_length">
                    </gdm>
                    <gdm type="segment_length">
                    </gdm>
                    <gdm type="default_grow_direction">
                    </gdm>
                    <gdm type="anastamosis_grow_direction">
                    </gdm>
                    <gdm type="branch_grow_direction">
                    </gdm>
               </grow_direction_modifiers>

               <boundary_condition type="bouncy">
                    <angio_boundary_groups>1</angio_boundary_groups>
                    <mbc type="pass_through"/>
               </boundary_condition>
          </common_properties>
     </material>
</Material>
```

References