(1.) We call fe_getforce in fe_mainExplicit. The goal is to calculate F_net. Note that F_net is called f_tot in all the subroutines. In general, $f^{net} = f^{ext} - f^{int}$. We know f^{ext}, so the real challenge is to find f^{int}. We find f^{int} as follows

one element domain (volume integral)

$$f^{int}_{I} = \int_{\Omega_0} B_{0I}^T \{S\} \, d\Omega_0 = \sum_{Q} B_{0I}^T \{S\} J_\xi^\circ \overline{W}_Q$$

f^{int}_I: 24×1 — nodal forces for one element

B_{0I}^T: 24×6 — $\{S\}$: 6×1

$\sum_Q$: sum over 8 quad points for a hex element

(2.) In fe_getforce, we go one of two ways depending on if embedded_constraint = true or not. This is a global bool variable that gets set to true in fe_mainRead if it finds a CONSTRAINT of type = embedded in the input file. Note that num_constraints is also a global variable that gets defined in fe_mainRead.

(3.) In this case, embedded_constraint = true, so our path is set. While still in fe_getforce, we identify host_id and embed_id by comparing master_name and slave_name (from the Constraint object) to mesh_name (from the Mesh objects). The id number is the element number in the array of Mesh objects created in fe_mainRead.

(4.) In fe_mainRead, we run the preprocess() method on each constraint. Within preprocess(), we repeat the above and identify host_id and embed_id. We then feed these into fe_embed_preprocessing, in order to find embed_map.

(5.) Finally, in fe_getforce, we call fe_getForce_3d_embed; where the goal is to calculate f_tot. Note that correct_vr is associated with the Constraint class bool object address_volume_redundancy; which is also a global variable. This variable is set in fe_mainRead, based on the input file.

(6.) We now move on to fe_getforce_3d_embed, which is the main function. Note that we are considering nine components for stress and strain vectors. For the majority of the function we are looping over the elements in the host Mesh object.

we start off by populating the xcoord, ycoord, zcoord 8x1 vectors for the element. We then use fe_gather_pbr, twice, to populate 24x1 vectors u_e and f_ext_e. Note that we need u_e to calculate F.

Note that we specify nglx=2, ngly=2, nglz=2; which are the number of quadrature points in the x, y, z directions. Therefore, there are <u>eight</u> quad points per element.

(7.) Now, we loop over each quad point in the element. First, we use fe_dniso_8 to find dndr, dnds, dndt, which are 8×1 vectors, at the quad point.

These are the components of $N_{,\xi}$. Note that the shape functions $N_I(\xi)$ are defined in the parent domain $\square$. Therefore, we can easily evaluate $N_{,\xi}$ for any point in $\square$.

Next, we use the chain rule to find $N_{,\mathbf{X}}$ at the quad point.

major goal #1

Jacobian, $\square \leftrightarrow \Omega_0$

$$\underset{8\times 3}{N_{,\mathbf{X}}} = \underset{8\times 3}{N_{,\xi}}\ \underset{3\times 3}{\mathbf{X}^{-1}_{,\xi}} = \underset{8\times 3}{N_{,\xi}}\ \underset{3\times 3}{\left(\mathbf{F}^{\circ}_{\xi}\right)^{-1}}$$

$$\underbrace{N_{,\mathbf{X}}}_{\mathbf{B}_{0I}^{T}} = \begin{bmatrix} \frac{\partial N_1}{\partial X} & \frac{\partial N_1}{\partial Y} & \frac{\partial N_1}{\partial Z} \\ \frac{\partial N_2}{\partial X} & \frac{\partial N_2}{\partial Y} & \frac{\partial N_2}{\partial Z} \\ \vdots & \vdots & \vdots \\ \frac{\partial N_8}{\partial X} & \frac{\partial N_8}{\partial Y} & \frac{\partial N_8}{\partial Z} \end{bmatrix}_{8\times 3}$$

Note:
The Jacobian does not vary with time. However, it is different for each element.

$$N_{,\xi} = \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & \frac{\partial N_1}{\partial \eta} & \frac{\partial N_1}{\partial \zeta} \\ \frac{\partial N_2}{\partial \xi} & \frac{\partial N_2}{\partial \eta} & \frac{\partial N_2}{\partial \zeta} \\ \vdots & \vdots & \vdots \\ \frac{\partial N_8}{\partial \xi} & \frac{\partial N_8}{\partial \eta} & \frac{\partial N_8}{\partial \zeta} \end{bmatrix}_{8\times 3}$$

$$\mathbf{X}_{,\xi} = \mathbf{F}^{\circ}_{\xi} = \begin{bmatrix} \frac{\partial X}{\partial \xi} & \frac{\partial X}{\partial \eta} & \frac{\partial X}{\partial \zeta} \\ \frac{\partial Y}{\partial \xi} & \frac{\partial Y}{\partial \eta} & \frac{\partial Y}{\partial \zeta} \\ \frac{\partial Z}{\partial \xi} & \frac{\partial Z}{\partial \eta} & \frac{\partial Z}{\partial \zeta} \end{bmatrix}_{3\times 3}$$

(Jacobian)

summation over nodes in element

$$\mathbb{X}_{i,\xi} = \mathbb{X}_{iI}\, N_{I,\xi}(\xi) \qquad \square \to \Omega_0$$

$\mathbb{X}_{iI}$: i = spatial component, I = node number

Note:

$N_{I,\xi}$ is easy to find. We use this, along with the nodal coordinates in Ω_0, to find the components of $\mathbb{X}_{,\xi}$. Then we invert $\mathbb{X}_{,\xi}$ and evaluate $N_{I,X}$, using the chain rule. We have to do this for <u>every</u> quad point in <u>each</u> element.

(8.) As outlined above, we use fe_calJacobian to calculate the 3×3 matrix jacobian. Then, we use fe_dndx_8_pbr to calculate the 8×1 vectors dndx, dndy, dndz.

(9.) Next, we use fe_strDispMatrix_totalLagrangian_pbr to calculate the 6×24 matrix disp_mat. Within this function, we use fe_calDefGrad_pbr to calculate the 3×3 vector F.

Important Note:

The strain-displacement matrix, B°_{jI} comes in two forms. You use the 3×8 form when solving for the gradient of the displacement field, H. You use the 6×24 Voigt form when solving for the 24×1 internal force vector, f^{int}_I.

major goal #2

identity matrix

$$\underset{3\times3}{F} = \underset{3\times3}{I} + \underset{3\times3}{H}$$

Note that you need the displacement in order to find F.

$$\underset{3\times3}{H} = \underset{3\times8}{u_I}\ \underset{8\times3}{B^{\circ T}_{oI}}$$

$$F_{ij} = \frac{\partial x_i}{\partial X_j} = \begin{bmatrix} \frac{\partial x}{\partial X} & \frac{\partial x}{\partial Y} & \frac{\partial x}{\partial Z} \\ \frac{\partial y}{\partial X} & \frac{\partial y}{\partial Y} & \frac{\partial y}{\partial Z} \\ \frac{\partial z}{\partial X} & \frac{\partial z}{\partial Y} & \frac{\partial z}{\partial Z} \end{bmatrix}$$

3×3

Note that we recently spent a lot of effort to find these components. We need them to find F and thus f_I^{int}.

$$u_I = \begin{bmatrix} u_{x1} & u_{x2} & \cdots & u_{x8} \\ u_{y1} & u_{y2} & \cdots & u_{y8} \\ u_{z1} & u_{z2} & \cdots & u_{z8} \end{bmatrix}$$

3×8

$$B_{0I}^T = \begin{bmatrix} \frac{\partial N_1}{\partial X} & \frac{\partial N_1}{\partial Y} & \frac{\partial N_1}{\partial Z} \\ \frac{\partial N_2}{\partial X} & \frac{\partial N_2}{\partial Y} & \frac{\partial N_2}{\partial Z} \\ \vdots & \vdots & \vdots \\ \frac{\partial N_8}{\partial X} & \frac{\partial N_8}{\partial Y} & \frac{\partial N_8}{\partial Z} \end{bmatrix}$$

8×3

major goal #3

$$B_I^o = \text{(Voigt)}$$

Note that these terms are the components of F.

$$\begin{bmatrix} \frac{\partial N_I}{\partial X}\frac{\partial x}{\partial X} & \cdots \\ \frac{\partial N_I}{\partial Y}\frac{\partial x}{\partial Y} & \cdots \\ \frac{\partial N_I}{\partial Z}\frac{\partial x}{\partial Z} & \cdots \\ \frac{\partial N_I}{\partial Y}\frac{\partial x}{\partial Z} + \frac{\partial N_I}{\partial Z}\frac{\partial x}{\partial Y} & \cdots \\ \frac{\partial N_I}{\partial X}\frac{\partial x}{\partial Z} + \frac{\partial N_I}{\partial Z}\frac{\partial x}{\partial X} & \cdots \\ \frac{\partial N_I}{\partial X}\frac{\partial x}{\partial Y} + \frac{\partial N_I}{\partial Y}\frac{\partial x}{\partial X} & \cdots \end{bmatrix}$$

6×24

(10.) Now, we use fe_stressUpdate_pbr to calculate the 6×1 vector sigma_e. This is the PK2 stress in Voigt notation. In fe_stressUpdate_pbr, we use fe_get_model to find the material model associated with the particular element. We then use this model to select the appropriate stress function. In this case, we will use fe_mooneyrivlin_hyperelastic_pbr to calculate sigma_e.

(11.) In fe_mooneyrivlin_hyperelastic_pbr, we start off by using the same fe_calDefGrad_pbr to calculate F. Once we have F, we can calculate C, I1, I2, defJacobian, I1_bar, I2_bar, C_bar. We then use fe_get_mats to obtain c1, c2, D; which are associated with the element's material number. We now calculate P, PK_S; where PK_S is a 3×3 matrix. Note that PK_S is the second Piola-Kirchoff stress (PK2); which is represented by S. Finally, we use fe_tensor2voigt to convert the 3×3 matrix PK_S to the 6×1 vector sigma_e; which is now in Voigt notation.

(12.) Now, back in fe_getForce_3d_embed, we calculate f_int_e. As previously mentioned,

$$\underset{24\times 1}{f^{int}_{I}} = \int_{\Omega_0} B_{0I}^{T} \{S\}\, d\Omega_0 = \sum_{Q} \underbrace{B_{0I}^{T}}_{24\times 6} \underbrace{\{S\}}_{6\times 1} \underbrace{J_{\xi}^{\circ}}_{\det(X,\xi)} \underbrace{\overline{w}_Q}_{w_{Q_1} w_{Q_2} w_{Q_3}}$$

$Q = 8$

where $X,\xi = F_{\xi}^{\circ}$ is the Jacobian $\square \leftrightarrow \Omega_0$

Note: $\det(A^T) = \det A$

Therefore, J_{ξ}° is the same for both versions of the Jacobian X,ξ.

Note: J_{ξ}° is different for each element.

This ends the loop over the quad points for the element. Next, we use fe_calCentroidStress_3d_pbr to calculate the 9x1 vector tmp_storage; which is the Cauchy stress at the centroid of the element. In fe_calCentroidStress_3d_pbr, we repeat all the steps we previously went through at each quad point, but this time at 0,0,0 in □. We end by using the same fe_stressUpdate_pbr. However, this time, we select return_opt = 1; which means, in fe_mooneyrivlin_hyperelastic_pbr, we calculate the Cauchy stress, not the PK2 stress. At the end of fe_calCentroidStress_3d_pbr, we manually convert the 6x1 vector sigma_centroid to the 9x1 vector element_stress; which is tmp_storage back in fe_getForce_3d_embed. Finally, we add the 9x1 vector tmp_storage to the (nel x 9) x 1 vector element_stress_host_local.

$$\sigma = J^{-1} F S F^{T}$$

- S: second Piola-Kirchoff (PK2) stress
- J: Jacobian determinant, $J = \det F$
- σ: Cauchy stress

Now, we use fe_calCentroidStrain_3d_pbr to calculate the 9x1 vector tmp_storage; which is the spatial logarithmic strain at the centroid of the element. Again, we repeat all the steps we previously went through to calculate F. We then use singular value decomposition to help find the polar decomposition of F; which we use to find lnV.

$$F = V R$$

- R: rotation tensor (orthogonal)
- V: spatial stretch tensor (symmetric)

singular values of A (diagonal matrix)

conjugate transpose

$$A = U S V^*$$

$V^* = V^T$, if V has real entries

spatial stretch tensor, not the same as V in the SVD

$$V = U S U^*$$

$$R = U V^*$$

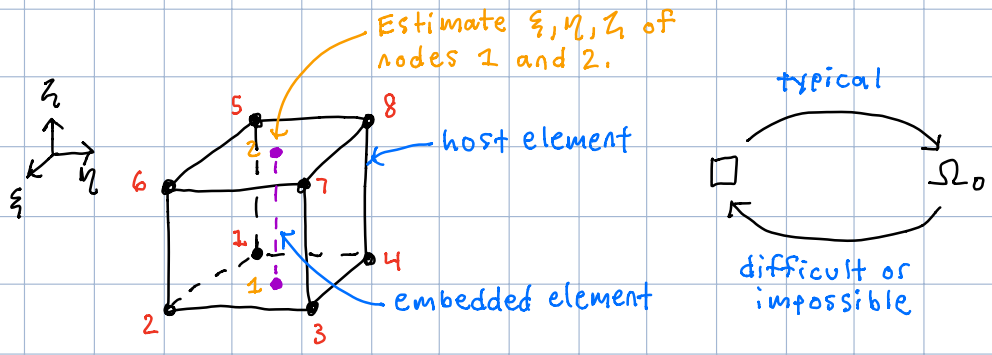
$$V = \sum_{\alpha=1}^{3} \lambda_\alpha \, n_\alpha \otimes n_\alpha$$

$$\ln V = \sum_{\alpha=1}^{3} \ln \lambda_\alpha \, n_\alpha \otimes n_\alpha$$

At the end of fe_calCentroidStrain_3d_pbr, we manually convert the 3x3 matrix E to the 9x1 vector element_strain; which is tmp_storage back in fe_getForce_3d_embed. Finally, we add the 9x1 vector tmp_storage to the (nel x 9) vector element_strain_host_local.

(13.) We now consider the embedded fibers. The main loop is over the rows of elements_embed; which contains all the elements in mesh[embed_id]. Recall that we are still in a loop over the elements of mesh[host_id]. Therefore, for each host element, we run a loop over all of the embedded elements. For each embedded element we use embed_map to see if the current host element number matches the host element id of the current embedded element. If we have a match, we then proceed.

we then loop over the columns of the embedded element row; which means we loop over the two node numbers that define the element. First, we extract the X, Y, Z coordinates of the node in Ω_o. Next, we use fe_newton Rhapson to find the 3x1 vector iso_coord_nodes; which are the estimated ξ, η, ζ coordinates of the node in $\square$.



$$X_i(\xi) = X_{iI}\, N_I(\xi) \qquad \square \rightarrow \Omega_o$$

Note:
Typically you have a point in $\square$. Then, using the X, Y, Z coordinates of the nodes (in Ω_o) and the shape functions, we can find the coordinates of the point in Ω_o. Here, we have to do the opposite.

(14.) Next, we calculate the 8x1 vector shapes, using the newly evaluated 3x1 vector iso_coord_nodes and fe_shapes_8. Then, we use fe_shape Matrix to calculate the 3x24 matrix shape_mat_embed. Now, we use shape_mat_embed and the 24x1 vector u_e to calculate the 3x1 displacement of fiber element node in Ω_o and insert it into the (no. of fiber nodes x 3) x 1 vector u_embed. We then copy the 3x1 displacement results into the 6x1 vector u_embed_local. Finally, we loop over the second node in the fiber element. Then, we continue on in the loop over the individual fiber elements.

Important Note:

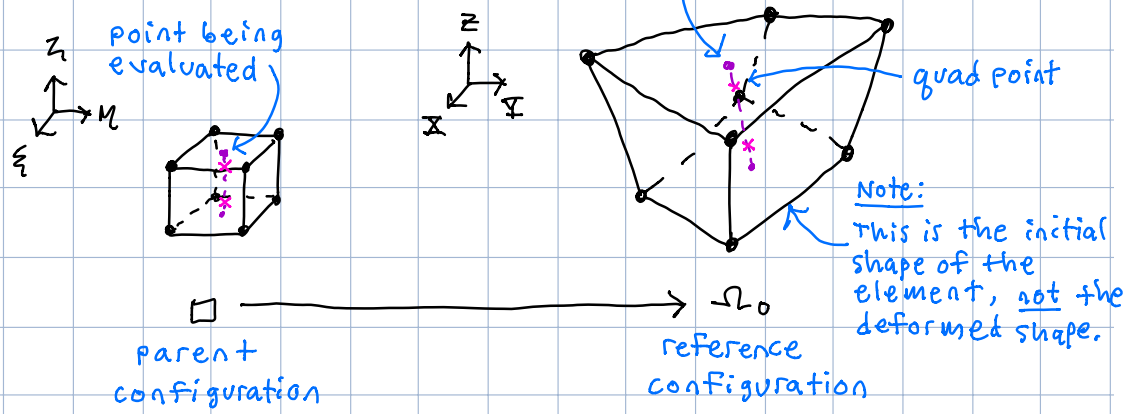
Recall that we know u_e going into fe_getForce_3d_embed; which are the nodal displacements of an individual host element.

displacement of fiber node in Ω_0 — location of fiber node in $\square$

$$\underset{\sim}{u}_i(\xi) = \underset{\sim}{u}_{iI} N_I(\xi) \qquad \square \to \Omega_0$$

$\underset{\sim}{u}_{iI}$: displacement of host element nodes

Note: we really just looked at what happens to a point within the host element.



(15.) During the previous loop, we stored the nodal coordinates (in Ω_0) of the fiber element in 2x1 vectors xcoord_embed, ycoord_embed, zcoord_embed. Now, we use fe_calVolume to calculate the length of the fiber element, length_embed, in Ω_0.

Pg. 356

$a_1 = x_2 - x_1$

$a_2 = y_2 - y_1$

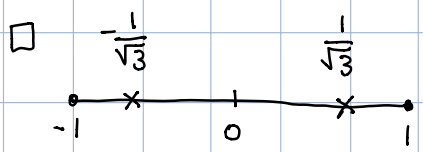
$a_3 = y_3 - y_3$

$\vec{a} = [a_1, a_2, a_3]$

$|\vec{a}| = \sqrt{a_1^2 + a_2^2 + a_3^2}$

(16.) Now, we enter a loop over the two quad points. We know the quad point locations in the 1D fiber parent domain. However, we need the quad point locations in the 3D host parent domain. First, we use fe_findIntgPoints_1d to calculate the 3x1 vector local_intg_points; which are the Ξ, Υ, Z coordinates of the quad point in Ω_0.

pg. 141



1D parent domain

This is basically a percentage of the overall $\Delta\Xi$.

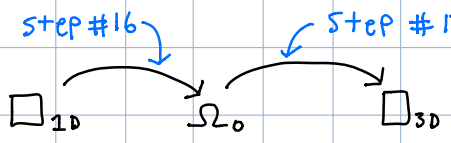
$$\Xi = \Xi_0 + \left(\frac{1 \pm \frac{1}{\sqrt{3}}}{2}\right)(\Xi_1 - \Xi_0)$$

$$\Upsilon = \Upsilon_0 + \left(\frac{1 \pm \frac{1}{\sqrt{3}}}{2}\right)(\Upsilon_1 - \Upsilon_0)$$

variable "point"

$$Z = Z_0 + \left(\frac{1 \pm \frac{1}{\sqrt{3}}}{2}\right)(Z_1 - Z_0)$$

(17.) Now, we use fe_newtonRhapson to calculate the 3x1 vector global_intg_points; which are the ξ, η, ζ coordinates of the quad point in the 3D host parent domain.



(18.) Now, we repeat the usual steps required to evaluate the 6x1 vector sigma_embed; which is the PK2 stress at the quad point. Note that we still using the same hex element formulas for the calculations. However, we are using the material number from the fiber element. Then, we calculate the internal force contribution, 24x1 vector f_int_truss, as follows:

area_truss (global variable)

length_embed (in Ω_0)

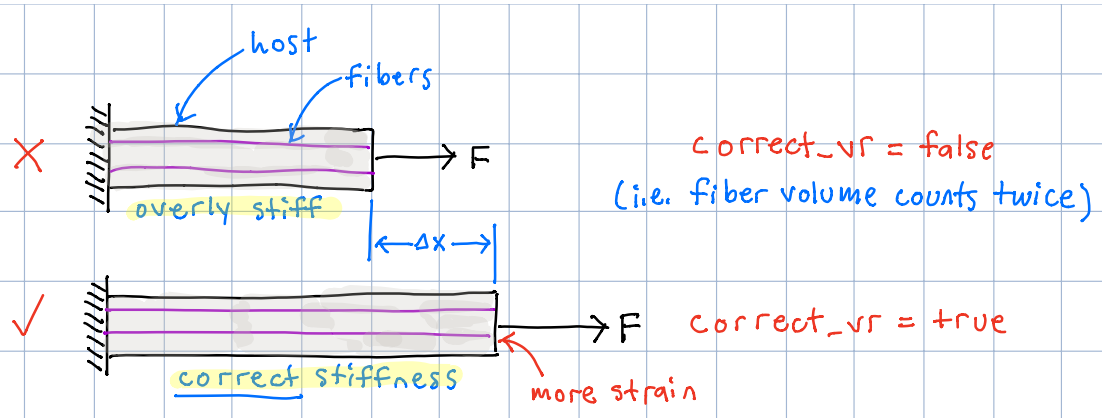
$$\underbrace{f_I^{int}}_{24\times 1} = \sum_Q \underbrace{B_{0I}^T}_{24\times 6} \underbrace{\{S\}}_{6\times 1} \overline{w}_Q A \frac{\ell}{2}$$

still using B_{0I}^T for hex element

Next, we add the fiber contribution f_int_truss to the original hex element internal force 24x1 vector f_int_e.

(19.) Back in fe_getForce, we extract the bool correct_vr from the Constraint object. We then pass correct_vr through fe_getForce_3d_embed. If correct_vr = true, we again use fe_stressUpdate_pbr to evaluate the 6x1 vector sigma_correction; which is the PK2 stress at the fiber quad point. This time we use the material number from the host element.

We then use the same formula as above to calculate the 24x1 vector f_int_correction. Next, we subtract this from f_int_e. This ends the loop over the fiber quad points.



correct_vr = false
(i.e. fiber volume counts twice)

correct_vr = true

(20.) Now, we are done looping over the two quad points in the specific fiber element. Next, we use fe_cal Centroid Strain _embed_3d_pbr to calculate the 9x1 vector tmp_storage; which is the spatial logarithmic strain of the specific fiber element. Note that the first element of element _strain is the only non-zero component. For reference, recall that u_embed_local is a 6x1 vector and xcoord _embed, ycoord_embed, zcoord_embed are 2x1 vectors.

$$\lambda = \frac{\ell}{L}$$

ℓ: final length

L: original length

$$\ln V = \ln \lambda \quad \text{(1D case)}$$

Next, we use fe_cal Centroid Stress_embed_3d_pbr to calculate the 9x1 vector tmp_storage; which is the Cauchy stress along the fiber. Note that we use the material number for the fiber. Also, recall that the 24x1 vector u_e is related to the host element in question. The 6x1 vector v_embed_ local is related to the fiber element in question. Finally, recall that the 8x1 vectors xcoord, ycoord, zcoord are related to the host element.

We start off by using fe_findIntgPoints_1d to find the 3x1 vector local_intg_points, which are the X, Y, Z coordinates of the fiber midpoint. We mapped $\xi = 0$ in $\square_{1D}$ to Ω_0. Next, we use fe_newtonRhapson to find the 3x1 vector global_intg_points, which are the estimated ξ, η, ζ coordinates of the fiber midpoint in $\square_{3D}$. We then go on to use the same fe_stressUpdate_pbr to calculate the 6x1 vector sigma_truss. As before, we indicate return_opt = 1; which means, in fe_mooneyrivlin_hyperelastic_pbr, we calculate the Cauchy stress, not the PK2 stress. Next, we use fe_voigt2tensor to convert the 6x1 vector sigma_truss to the 3x3 matrix temp_stress.

At this point, we have the Cauchy stress tensor at the centroid of the fiber. However, the stress tensor components correspond to the global coordinate system in Ω_0. We want the stress along the fiber, so we need to complete a transformation of axes. Note that the state of stress remains the same; it is the components that change.

pg. 32

$$[S]' = [Q]^T [S] [Q]$$

$[Q]$: transformation matrix

$[S]$: second order tensor

where

$$[Q] = \begin{bmatrix} e_1 \cdot e_1' & e_1 \cdot e_2' & e_1 \cdot e_3' \\ e_2 \cdot e_1' & e_2 \cdot e_2' & e_2 \cdot e_3' \\ e_3 \cdot e_1' & e_3 \cdot e_2' & e_3 \cdot e_3' \end{bmatrix}$$

(directional cosines)

$$e_3 \cdot e_2' = \underbrace{|e_3|}_{1} \underbrace{|e_2'|}_{1} \cos\theta_{32'} = \cos\theta_{32'}$$

We start of by using fe_calTransformation to calculate the 3x3 matrix T. Note that we specify the format of T using choice = 3. The function uses the 2x1 vectors xcoord_embed, ycoord_embed, zcoord_embed; which are the fiber node coordinates in Ω_0.

First, we establish the new coordinate system as the 3x1 vectors dir_truss_x, dir_truss_y, dir_truss_z. Then, we normalize them, in order to make them unit vectors. Next, we establish the reference configuration axes as the 3x1 vectors dir_global_x, dir_global_y, dir_global_z. Then, we define the directional cosines as follows:

$$l_1 = \vec{X} \cdot \vec{X}', \quad l_2 = \vec{X} \cdot \vec{Y}', \quad l_3 = \vec{X} \cdot \vec{Z}'$$

$$m_1 = \vec{Y} \cdot \vec{X}', \quad m_2 = \vec{Y} \cdot \vec{Y}', \quad m_3 = \vec{Y} \cdot \vec{Z}'$$

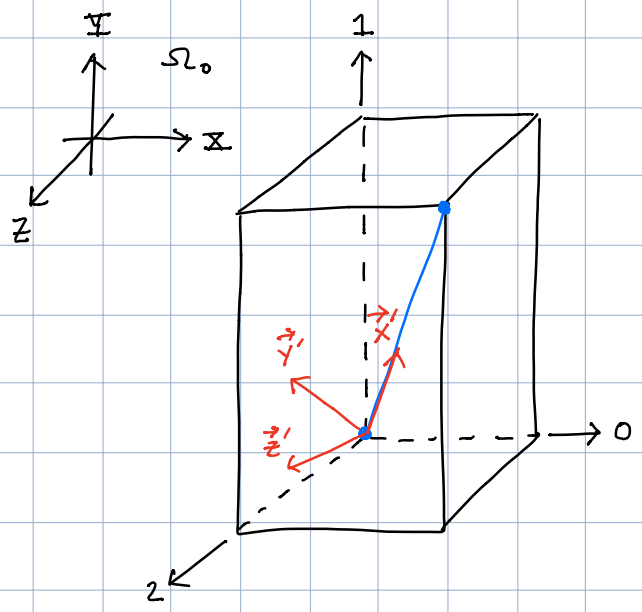
$$n_1 = \vec{Z} \cdot \vec{X}', \quad n_1 = \vec{Z} \cdot \vec{Y}', \quad n_3 = \vec{Z} \cdot \vec{Z}'$$

Finally, using choice = 3, we define T as follows:

$$T = \begin{bmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ n_1 & n_2 & n_3 \end{bmatrix}$$

For reference, dir_truss_x, dir_truss_y, dir_truss_z we created using the following logic.

$$x_0' = x_1 - x_0$$
$$x_1' = y_1 - y_0$$
$$x_2' = z_1 - z_0$$



- If $x_0' \neq 0$

$$y_0' = -\left[\frac{y_1' x_1' + y_2' x_2'}{x_0'}\right]$$
$$y_1' = 1$$
$$y_2' = 0$$

- If $x_0' = 0$ and $x_1' \neq 0$

$$y_0' = 1$$
$$y_1' = -\left[\frac{y_2' x_2' + y_0' x_0}{x_1'}\right]$$
$$y_2' = 1$$

- If $x_0' = 0$ and $x_2' \neq 0$

$$y_0' = 1$$
$$y_1' = 0$$
$$y_2' = -\left[\frac{y_0' x_0' + y_1' x_1'}{x_2'}\right]$$

$$\vec{z}' = \vec{x}' \times \vec{y}'$$

Now, back in fe_calCentroidStress_embed_3d_pbr, we use T to carry out a coordinate transformation on temp_stress; which is the Cauchy stress at the midpoint of the fiber.

Note that we only pass temp_stress(0,0) back to the 9x1 vector tmp_storage in fe_getForce_3d_embed. This ends the loop over fibers. However, we are still in the same host element.

Now, we calculate the 24x1 vector f_tot_e as follows:

f_tot_e = f_ext_e - f_int_e

Then, we use fe_scatter_pbr to scatter f_tot_e to the Sdof x 1 vector f_tot in fe_getforce; which is the vector F_net in fe_mainExplicit. This ends the loop over host elements.