Implementation of Finite Deformation Capability in the `imitor` Finite Element Code

Omar Hafez

ECI 289E: Advanced Finite Element Methods

Introduction

Implementation of finite deformations in a finite element code involves several considerations that are not needed for the small deformation case. In addition to the machinery for small deformations, a kinematic algorithm is required to decide the manner in which to perform stretching and rotating through a time step given an incremental deformation gradient. Through this process, incremental strain rates and now rotations as well are computed to perform the Cauchy stress update from one time step to the next. The residual equations are generally different for problems with finite deformations, involving first Piola-Kirchhoff stress rather than Cauchy stress. Thus, solution techniques of the nonlinear residual equations require computation of additional terms and their derivatives that are not needed for small deformations.

This document outlines the calculations and considerations performed to implement finite deformations into the object-oriented finite element code `imitor`. This involves a description and derivation of the kinematic algorithm and the steps taken for solution of the residual equations, followed by a description of the source code, and finally a description of some verification and demonstration problems.

Kinematic Algorithm

The following kinematic algorithm takes as input an incremental deformation gradient and outputs a constant rate-of-stretching tensor and a rotation tensor. It follows the steps outlined in a 1993 paper by Rashid.[1] The following discussion will correspond to computations for a particular element e.

First define the position of a material at the end of time t_n as $\overline{\mathbf{x}}$, the position at time t_{n+1} as $\mathbf{x}$, and the position in a reference configuration as $\mathbf{X}$. Define displacements $\mathbf{U} = \overline{\mathbf{x}} - \mathbf{X}$, $\hat{\mathbf{u}} = \mathbf{x} - \overline{\mathbf{x}}$, and $\mathbf{u} = \mathbf{x} - \mathbf{X}$. For conventional finite elements, we can define the displacements and their gradients for a given element as a linear combination of shape functions ϕ:

$$u_i = u_{ia}\phi_a, \qquad U_i = U_{ia}\phi_a, \qquad \hat{u}_i = \hat{u}_{ia}\phi_a \tag{1}$$

$$u_{i,j} = u_{ia}\phi_{a,j}, \qquad U_{i,j} = U_{ia}\phi_{a,j}, \qquad \hat{u}_{i,j} = \hat{u}_{ia}\phi_{a,j} \tag{2}$$

where repeated indices follow the summation convention. In what follows, indices a and b will be reserved to range over the nodes, whereas other indices will range over the spatial dimension of the problem, either 2 or 3. Note above that when displacement u has two indices, it corresponds to nodal displacements. The notation for derivatives $\frac{\partial(\cdot)_i}{\partial X_j} = (\cdot)_{i,j}$ is used. The following implementation is peformed in a total Lagrangian sense, meaning all nodes and integration points remain fixed to material points in the reference configuration

[1] Rashid, M. M. (1993), Incremental kinematics for finite element applications. Int. J. Numer. Meth. Engng., 36: 3937 - 3956

for all time. Thus, derivatives described above are always with respect to the reference configuration coordinates. The deformation gradients of interest are defined as:

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}}, \qquad \overline{\mathbf{F}} = \frac{\partial \overline{\mathbf{x}}}{\partial \mathbf{X}}, \qquad \hat{\mathbf{F}} = \frac{\partial \mathbf{x}}{\partial \overline{\mathbf{x}}} \tag{3}$$

Thus, $\mathbf{F}$ can be decomposed as

$$\mathbf{F} = \hat{\mathbf{F}}\overline{\mathbf{F}} \tag{4}$$

It can be shown that the deformation gradients can be written in terms of the nodal displacements and shape function gradients in the following form:

$$F_{ij} = \delta_{ij} + (U_{ia} + \hat{u}_{ia})\,\phi_{a,j} = \delta_{ij} + (U_{ia} + \hat{u}_{ia})\,B_{ja} \tag{5}$$

$$\overline{F}_{ij} = \delta_{ij} + U_{ia}\phi_{a,j} = \delta_{ij} + U_{ia}B_{ja} \tag{6}$$

where B_{ja} are the components of the shape function gradient for node a and element e, evaluated at $t = t_{n+1}$, and δ_{ij} are the components of the Kronecker delta function.

To avoid floating point precision issues, we define $\mathbf{A} = \mathbf{I} - \hat{\mathbf{F}}^{-1}$, where $\mathbf{I}$ is the identity tensor. Once $\mathbf{F}$ and $\overline{\mathbf{F}}$ are formed, $\mathbf{A}$ can be formed as $\mathbf{A} = \mathbf{I} - \overline{\mathbf{F}}\mathbf{F}^{-1}$.

The algorithm essentially assumes a motion involving a constant rate-of-stretch tensor $\mathbf{D}$ that in the absence of any rotation, produces $\hat{\mathbf{U}}$, where $\hat{\mathbf{F}} = \hat{\mathbf{R}}\hat{\mathbf{U}}$. This stretch is followed by the pure rotation $\hat{\mathbf{R}}$.

Define the velocity gradient $\mathbf{L} = \frac{\partial \mathbf{u}}{\partial \mathbf{x}} = \dot{\mathbf{F}}\mathbf{F}^{-1}$. The velocity gradient can be decomposed into symmetric and anti-symmetric parts $\mathbf{D}$ and $\mathbf{W}$, respectively, where $\mathbf{D} = \frac{1}{2}\left(\mathbf{L} + \mathbf{L}^T\right)$ and $\mathbf{W} = \frac{1}{2}\left(\mathbf{L} - \mathbf{L}^T\right)$. In the absence of rotation, $\dot{\mathbf{F}}\mathbf{F}^{-1} = \dot{\mathbf{U}}\mathbf{U}^{-1}$ and $\mathbf{L} = \mathbf{D}$. Combining results yields $\dot{\mathbf{U}}\mathbf{U}^{-1} = \mathbf{D}$. The calculation of the rate-of-stretch tensor involves solving the following ODE:

$$\dot{\mathbf{U}} = \mathbf{D}\mathbf{U}, \qquad t \in \{t_n, t_{n+1}\} \tag{7}$$

$$\mathbf{U}\left(t_n\right) = \mathbf{I} \tag{8}$$

The solution of this ODE yields an exponential solution. Imposing the desired $\mathbf{U}\left(t_{n+1}\right) = \hat{\mathbf{U}}$ yields $\hat{\mathbf{U}} = exp\left(\mathbf{D}\Delta\right)$, where $\Delta = t_{n+1} - t_n$. Performing a Taylor expansion about $\hat{\mathbf{U}} = \mathbf{I}$ and making use of the right Cauchy-Green deformation tensor $\hat{\mathbf{C}} = \hat{\mathbf{U}}^2$ yields the following result:

$$\mathbf{D}\Delta \approx -\frac{1}{2}(\hat{\mathbf{C}}^{-1} - \mathbf{I}) + \frac{1}{4}(\hat{\mathbf{C}}^{-1} - \mathbf{I})^2 - \frac{1}{6}(\hat{\mathbf{C}}^{-1} - \mathbf{I})^3 \tag{9}$$

The assumption that $\hat{\mathbf{U}}$ is near $\mathbf{I}$ is valid as long as the time steps are chosen that the stretch in each time step is relatively small. As mentioned above, because $\hat{\mathbf{C}}^{-1}$ is close to identity, to avoid floating point precision issues, rather than forming $\hat{\mathbf{C}}^{-1}$ directly from $\hat{\mathbf{F}}^{-1}$, we compute the strain increment as follows:

$$\hat{\mathbf{C}}^{-1} - \mathbf{I} = \mathbf{A}\mathbf{A}^{\mathbf{T}} - \mathbf{A} - \mathbf{A}^{\mathbf{T}} \tag{10}$$

$$\mathbf{D}\Delta = -\frac{1}{2}(\hat{\mathbf{C}}^{-1} - \mathbf{I}) + \frac{1}{4}(\hat{\mathbf{C}}^{-1} - \mathbf{I})^2 - \frac{1}{6}(\hat{\mathbf{C}}^{-1} - \mathbf{I})^3 \tag{11}$$

With the strain increment now formed, what remains is the rotation increment. The calculation of the rotation tensor $\mathbf{R}$ is approximated, but nonetheless as a proper orthogonal tensor. Please refer to the aformentioned Rashid paper for a complete description. The result is shown below:

$$\hat{R}_{ij} = \delta_{ij}\cos\theta + \frac{1-\cos\theta}{4Q}\alpha_i\alpha_j - \frac{\sin\theta}{2\sqrt{Q}}\varepsilon_{ijk}\alpha_k \tag{12}$$

where θ is the rotation angle, $\boldsymbol{\alpha}$ is related to the rotation axis, $\boldsymbol{\varepsilon}$ is the alternator symbol, and P and Q are quantities used for the polynomial approximation of $\cos^2\theta$. Note, special case is taken for Q close to zero, as outlined in the paper.

The typical Cauchy stress evolution equation is $\overset{\circ}{\mathbf{T}} = \mathbf{C}(\mathbf{D}-\mathbf{D}^p)$, where $\overset{\circ}{\mathbf{T}} = \dot{\mathbf{T}}+\mathbf{TW}-\mathbf{WT}$ is the Jaumann rate of stress, $\mathbf{C}$ is the isotropic rank-four material modulus tensor, and $\mathbf{D}^p$ is the plastic deformation rate. For the purposes of this document, we assume $\mathbf{D}^p = \mathbf{0}$, i.e., that we are dealing only with elastic materials. For the proposed kinematic algorithm, there is no vorticity during the stretch, so the rate equation simplifies to $\dot{\mathbf{T}} = \mathbf{C}(\mathbf{D}-\mathbf{D}^p)$. Thus, $\mathbf{D}$ is calculated above, fed into the constitutive model to update the Cauchy stress to an unrotated state $\hat{\mathbf{T}} = \mathbf{CD}\Delta$, and then finally the end step Cauchy stress is forward rotated by $\hat{\mathbf{R}}$:

$$\mathbf{T} = \hat{\mathbf{R}}\hat{\mathbf{T}}\hat{\mathbf{R}}^T \tag{13}$$

Thus, the incremental strains and rotations given an incremental deformation gradient have been defined.

Solution of the Nonlinear Residual Equations

The residual equation for a particular element e is as follows:

$$R^e_{ia} = \int_{\Omega_e} P_{ij}\phi_{a,j}dv - \int_{\Omega_e}\rho_0 b_i\phi_a dv - \int_{\partial_t\Omega_e}\overline{p}_i\phi_a da \tag{14}$$

where Ω_e is the domain of the element and $\partial_t\Omega_e$ is the portion of its boundary where tractions are specified. The first Piola-Kirchhoff stress tensor is defined as $\mathbf{P} = J\mathbf{T}\mathbf{F}^{-T}$, where $J = det(\mathbf{T})$. The quantities $\phi_{a,j}$ are the element shape function gradients, ρ_0 is the density of the material in the reference configuration, $\mathbf{b}$ is the body force (ignored for our purposes), and $\overline{\mathbf{p}}$ the prescribed Piola traction on the traction boundary. Once again, the derivatives are with respect to the reference coordinate frame. Performing Newton-Raphson for the solution of $R^e_{ia} = 0$ yields:

$$\left.\frac{\partial R^e_{ia}}{\partial\hat{u}_{jb}}\right|_k \delta\hat{u}_{jb} = -\left.R_{ia}\right|_k \tag{15}$$

where k is the iteration number and $\delta\hat{u}_{jb}$ is the correction to the incremental nodal displacement.

An integration point's contribution to this equation for element e shows up on both sides, in calculating the element tangent stiffness, and in forming the forcing vector.

Calculating the contribution to the forcing vector is relatively trivial. For the first term, one must simply calculate $P_{ij} = JT_{ik}F^{-1}_{jk}$ and then form the integrand, where T_{ik} has already been calculated at this point. Elsewhere in `imitor`, integrand values at integration points are used with their weights to calculate an approximation to the integral using Gauss quadrature. For our purposes, there is no body force, so the second term does not need to be formed. Because the code is currently written so that Piola tractions are specified, the input tractions are simply multiplied by shape functions to compute the integrand of the third term. Those integrals are computed using Gauss quadrature over the appropriate area then and summed with the contribution from the first term.

If the solution pass requires more than just an evaluation of the residual, the element tangent stiffness must also be calculated. For a constant reference density, and assuming body force is not a function of displacement, the tangent stiffness is:

$$\frac{\partial R^e_{ia}}{\partial \hat{u}_{jb}} = \int_{\Omega_e} \frac{\partial P_{ik}}{\partial \hat{u}_{jb}} \phi_{a,k} dv - \int_{\partial_t \Omega_e} \frac{\partial \overline{p}_i}{\partial \hat{u}_{jb}} \phi_a da \tag{16}$$

Note, because we are using a total Lagrangian formulation, the derivatives can go inside the integral, and the shape functions are constant with respect to incremental nodal displacements. When Piola tractions are specified, the second term is zero, although if Cauchy tractions were specified, that term would contribute as well, since Cauchy tractions are defined over an area that depends on incremental nodal displacements.

Using the product rule,

$$\frac{\partial \mathbf{P}}{\partial \hat{\mathbf{u}}} = \frac{\partial J}{\partial \hat{\mathbf{u}}} \mathbf{T}\mathbf{F}^{-T} + J \frac{\partial \mathbf{T}}{\partial \hat{\mathbf{u}}} \mathbf{F}^{-T} + J\mathbf{T} \frac{\partial \mathbf{F}}{\partial \hat{\mathbf{u}}}^{-T} \tag{17}$$

The Cauchy stress will already have been provided, and calculating $\mathbf{F}^{-T}$ essentially involves calculating the inverse of the definition of $\mathbf{F}$ above and taking care for how a transpose tensor operates. Thus, the terms of interest are $\frac{\partial J}{\partial \hat{\mathbf{u}}}$, $\frac{\partial \mathbf{T}}{\partial \hat{\mathbf{u}}}$, and $\frac{\partial \mathbf{F}}{\partial \hat{\mathbf{u}}}^{-T}$. Provided in the `derivations` directory is a file titled `derivations.pdf`. Although admittedly messy, all necessary derivations to perform these calculations are written out in that document. Provided below are all of the useful intermediate results that were calculated to form the derivative of the first Piola-Kirchoff tensor with respect to incremental nodal displacements.

$$\frac{\partial J}{\partial F_{kl}} = JF^{-1}_{lk} \tag{18}$$

$$\frac{\partial F_{ij}}{\partial \hat{u}_{ka}} = \delta_{ik}\delta_{pa}B_{jp} \tag{19}$$

$$\frac{\partial J}{\partial \hat{u}_{ka}} = JF^{-1}_{pk}B_{pa} \tag{20}$$

$$\frac{\partial F^{-1}_{ij}}{\partial F_{kl}} = -F^{-1}_{ik}F^{-1}_{lj} \tag{21}$$

$$\frac{\partial F^{-1}_{ij}}{\partial \hat{u}_{ka}} = -F^{-1}_{ik}B_{pa}F^{-1}_{pj} \tag{22}$$

$$A_{ij} = \hat{u}_{ia} B_{ka} F^{-1}_{kj} \tag{23}$$

$$\hat{\mathbf{C}^{-1}} = \mathbf{I} - \mathbf{A} - \mathbf{A}^T + \mathbf{A}\mathbf{A}^T \tag{24}$$

$$\frac{\partial A_{ij}}{\partial \hat{u}_{ka}} = \hat{F}^{-1}_{ik} B_{pa} F^{-1}_{pj} \tag{25}$$

$$\frac{\partial \hat{C}^{-1}_{ij}}{\partial \hat{A}_{kl}} = -\delta_{ik}\delta_{jl} - \delta_{jk}\delta_{il} + \delta_{ik}A_{jl} + A_{il}\delta_{jk} \tag{26}$$

$$\begin{aligned}\frac{\partial(\mathbf{D}\Delta)}{\partial \mathbf{A}} =& -\frac{1}{2}\frac{\partial \hat{\mathbf{C}}^{-1}}{\partial \mathbf{A}} + \frac{1}{4}(\hat{\mathbf{C}}^{-1} - \mathbf{I})\frac{\partial \hat{\mathbf{C}}^{-1}}{\partial \mathbf{A}} + \frac{1}{4}\frac{\partial \hat{\mathbf{C}}^{-1}}{\partial \mathbf{A}}(\hat{\mathbf{C}}^{-1} - \mathbf{I}) \\ &- \frac{1}{6}\frac{\partial \hat{\mathbf{C}}^{-1}}{\partial \mathbf{A}}(\hat{\mathbf{C}}^{-1} - \mathbf{I})^2 - \frac{1}{6}(\hat{\mathbf{C}}^{-1} - \mathbf{I})\frac{\partial \hat{\mathbf{C}}^{-1}}{\partial \mathbf{A}}(\hat{\mathbf{C}}^{-1} - \mathbf{I}) - \frac{1}{6}(\hat{\mathbf{C}}^{-1} - \mathbf{I})^2\frac{\partial \hat{\mathbf{C}}^{-1}}{\partial \mathbf{A}}\end{aligned} \tag{27}$$

$$\frac{\partial(D_{ij}\Delta)}{\partial \hat{u}_{ka}} = \frac{\partial(D_{ij}\Delta)}{\partial A_{mn}}\frac{\partial A_{mn}}{\partial \hat{u}_{ka}} \tag{28}$$

$$\frac{\partial \hat{R}_{ij}}{\partial A_{mn}} = \frac{1}{2}(-\delta_{in}\delta_{jm} + \hat{R}_{im}\hat{R}_{nj}) \tag{29}$$

$$\frac{\partial \hat{R}_{ij}}{\partial \hat{u}_{ka}} = \frac{1}{2}\left(-\hat{F}^{-1}_{jk}(F^{-1}_{pi}B_{pa}) + \hat{R}_{im}\hat{F}^{-1}_{mk}\hat{R}_{nj}(F^{-1}_{pn}B_{pa})\right) \tag{30}$$

$$\frac{\partial \hat{\mathbf{T}}}{\partial \hat{\mathbf{u}}} = \frac{\partial \hat{\mathbf{T}}}{\partial(\mathbf{D}\Delta)}\frac{\partial(\mathbf{D}\Delta)}{\partial \hat{\mathbf{u}}} \tag{31}$$

$$\frac{\partial \mathbf{T}}{\partial \hat{\mathbf{u}}} = \frac{\partial \hat{\mathbf{R}}}{\partial \hat{\mathbf{u}}}\hat{\mathbf{T}}\hat{\mathbf{R}}^T + \hat{\mathbf{R}}\frac{\partial \hat{\mathbf{T}}}{\partial \hat{\mathbf{u}}}\hat{\mathbf{R}}^T + \hat{\mathbf{R}}\hat{\mathbf{T}}\frac{\partial \hat{\mathbf{R}}^T}{\partial \hat{\mathbf{u}}} \tag{32}$$

Note $\partial \hat{R}_{ij}/\partial A_{mn}$ makes use of the assumption that $\hat{\mathbf{U}} \approx \mathbf{I}$.

Care must be taken to calculate the double contraction of $\partial \hat{\mathbf{T}}/\partial(\mathbf{D}\Delta)$ with $\partial(\mathbf{D}\Delta)/\partial \hat{\mathbf{u}}$ correctly for calculation of $\partial \hat{\mathbf{T}}/\partial \hat{\mathbf{u}}$. Specifically, if the shear terms in $\mathbf{D}$ are stored as D_{12}, D_{13}, and D_{23}, then a factor of 2 shows up in the double contraction due to symmetry.

If the material is hypoelastic, $\hat{\mathbf{T}} = \overline{\mathbf{T}} + \mathbf{C}\mathbf{D}\Delta$, where $\overline{\mathbf{T}}$ is the beginning step stress. In this stress update as well, care must be made for the extra factor of 2 for the double contraction of $\mathbf{C}$ and $(\mathbf{D}\Delta)$. These are the only two places where an additional factor of 2 would be needed. In the following description, the tangent modulus $\partial \hat{\mathbf{T}}/\partial(\mathbf{D}\Delta)$ should not be modified. However, it must be calculated correctly.

For a hypoelastic material, $\dot{\mathbf{T}} = \mathbf{C}\mathbf{D}$. This can be written as $\dot{T_{ij}} = \lambda\delta_{ij}D_{kk} + 2\mu D_{ij}$. This 2 is precisely coming from the double contraction of $\mathbf{C}$ and $\mathbf{D}$. To get $\partial T_{ij}/\partial(D_{kl}\Delta)$, one must take *special* care to realize that, for a non-symmetric tensor, $\partial A_{ij}/\partial A_{kl} = \delta_{ik}\delta_{jl}$, but for a *symmetric*, $\partial A_{ij}/\partial A_{kl} = \frac{1}{2}(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk})$. *These are not the same* - this special consideration for symmetric tensors must be used. Then, $\partial T_{ij}/\partial(D_{kl}\Delta) = \lambda\delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk})$, which is precisely the definition for C_{ijkl}. Thus, when carefully considering the additional factors of 2 in double contractions, and considering the derivative of a tensor with itself has a different definition for symmetric tensors, we get that $\partial \hat{\mathbf{T}}/\partial(\mathbf{D}\Delta) = \mathbf{C}$ for a hypoelastic material, and the quantities of interes will be calculated correctly.

With the derivatives provided above, the stiffness contribution from the integration points of each element can be formed.

Source Code

The source code is provided as part of the `imitor_hafez.tar.gz` tarball in the directory `imitor_hafez`. Three files are edited: `elems_nodes_m`, `ip_strain_inc_m.f`, and `ip_frc_stf_m.f`.

The file `elems_nodes_m.f` is edited such that in the event of finite deformations, the subroutines `STRAIN_INC_LRG_DEF`, `FWD_ROT`, and `IP_CONT_LRG_DEF` are called, and the variable `ROT_INC` is allocated (and deallocated).

The file `ip_strain_inc_m.f` includes the subroutine `STRAIN_INC_LRG_DEF`, that calculates the strain increment $\mathbf{D}\Delta$ and the rotation increment $\hat{\mathbf{R}}$. The strain increment is stored as 6×1 array, utilizing symmetry, whereas the rotation increment is stored as a 3×3 array.

The file `ip_frc_stf_m.f` is modified in three main locations. First, the subroutine `FWD_ROT` forward rotates the Cauchy stress that is output from the material model, using the relation $\mathbf{T} = \hat{\mathbf{R}}\hat{\mathbf{T}}\hat{\mathbf{R}}^T$, as per the above discussion regarding an appropriate kinematic update. A map `IJ` from tensor indices to column-vector indices is utilized here as well as several times in the subroutine `IP_CONT_LRG_DEF` for symmetric tensors. Second, the subroutine `IP_CONT_LRG_DEF` follows the steps laid out in the previous section to populating the stiffness and residual for each integration point. Symmetries in the stress tensors and tangent modulus are using in conjunction with the tensor index map to reduce storage as well as computations.

The module `matrix_m.f` is `USE`d in both `ip_strain_inc_m.f` and `ip_frc_stf_m.f` for convenience. Arrays are sized assuming a three-dimensional problem, and are operated on accordingly. The computational time saved would have been minimal had arrays been sized specifically for two-dimensional problems in the case of 2D. In general, care is taken to minimize floating-point operations (FLOPs) and computational time through the use of FORTRAN's vectorized operation capability, calculating intermediate quantities, and the use of symmetry in the tensors. Coupled thermal problems are not considered for this extension to finite deformations, although they could be added with minimal change to the current code. The code is commented well enough that it should be quite clear the additions and modifications that are made.

Verification and Demonstration Problems

Verification and demonstration problems are located in the directory `test_probs`. All test problem names are suffixed `_sm` or `_lrg` depending on whether the small deformation formulation or the finite deformation formulation was used to solve the problem. Each test problem is run using both the small and finite deformation formulations.The directory entitled `mark` includes one-element traction and displacement BC problems, as well as nine-element patch tests with traction and displacement BCs as well. These four problems were run using the finite deformation formulation, and match the results given from small deformations. This

confirms at the most basic level that to zeroth order, the finite deformation formulation was implemented correctly.

In directory `omar`, there are four tests. Test 1 involves applying a displacement BC on the patch test mesh that results in 50% strains. Because there are no rotations, the Cauchy stress rate equation is integrated to produce exact results. The results from the finite deformation simulation match the exact results. Please refer to the last two pages of `derivations.pdf` for the derivation of the exact solution. Care was taken to consider the exact Poisson effect, as opposed to simply using the results from plane strain problems in small strain linear elasticity. These results give confidence that the strain increment and strain increment derivatives were calculated correctly for the diagonal terms.

Test 2 involves now applying tractions on the patch test mesh to produce roughly 50% strains. Qualitatively speaking, the stresses turn out to be higher for the finite deformation case than the small deformation case. For larger displacements, the area over which the Piola tractions act reduces, and the stresses go higher, whereas that effect does not come into play for small deformations. Considering this qualitative observation, it also makes intuitive sense that in Test 1 the stress were lower for the finite deformation case than for the small deformation case. This test showed that at least qualitatively the tractions are being applied correctly.

The next test performed was `b1` in the `brian` directory. This problem involves pure rotation with no stress on a single element. This was used with permission from Mr. Brian Giffin. The element rotates 90°clockwise with minimal stress successfully for the finite deformation case, whereas the small deformation case gives meaningless displacement results. This confirmed that the rotation increment was being calculated correctly.

The following test performed was Test 3 from `omar`. This test has small stretch and large rotation. It essentially reproduces the results from the Rashid paper mentioned above, with the difference that the mesh used here is 2D, resulting in a plane strain problem. The mesh is four elements long and one element thick. An automatic mesh generator s provided in the directory `omar/meshgen`. This generator produces the necessary time functions to produce the displacements that result from the deformation gradient provided. Through a 180° rotation, the final stresses reproduce the same stresses in the absence of rotation. Further, the results once again agree with the exact solution. Again, care was taken not to use Poisson effect from small deformations, but rather enforce the stress in the nonloading direction to be zero directly from the results of the stress rate equations. Interestingly, the results were not able to converge for very small time steps. The rotation through 180° required a partition of 24 steps, with on the order of 25 iterations to converge every step. In the Rashid paper, however, the time interval is only partitioned into six steps. It is not clear at this point whether the code is converging slower than it should be.

The next text was `b2` in the `brian` directory. Again, this problem was designed by Giffin. It involves a stretch and a rotation, and is not a proportional loading scheme. Although some additional interpretation and reasoning of the results is certainly possible, the problem was run more as a verification between two different codes, which has its own utility

for verification purposes. Indeed, the results match those of both Giffin and Mr. Steven Wopschall.

The final test problem involves large shear, large stretch, and large rotation, resulting in a nonuniform stress field. It is likely possible to integrate the rate equations to get exact solutions for reference even for problems with vorticity. However, to this point a shear problem is presented as Test 4 in `omar` merely for demonstration purposes. The body is a unit square, and the mesh is a uniform 5x5. A traction is applied along the top directed to the right, and the bottom of the body is completely fixed. This problem is essentially a cantilever problem, except the geometry of the cantilever certainly does not match that of Euler-Bernoulli beam theory, and the deformations imposed certainly are not accommodated by small strain linear elasticity. The body rotates about 30°, and there is significant stretching and shearing. Qualitatively, the results make intuitive sense. The left side of the body is in tension, and the right side of the body in compression. Along any horizontal plane, the left half displaces upwards, and the right half displaces downwards. Similarly, roughly the left half of the body is tension for T_{22}, and the right half of the body is in compression. There is up to a 20% error in the displacement had the problem been modeled as small deformation.

A side note in running these simulations, interestingly truncating the definition of the strain increment from two terms to one term resulted in roughly a 5% difference in stresses for the `b2` test problem.

Future Steps

Obviously there is much to continue working on from this point. The first will be to allow for Cauchy tractions to be specified. Secondly, verification testing will continue, hopefully confirming convergence rates along the way. An exact solution problem with shearing would be useful for comparison. Extending to three dimensions, implementing hyperelastic materials and plastic deformation, and adding different element types are all obvious extensions as well.