OpenCMISS Learning Documentation

# Introduction

This document is intended to be a place where the OpenCMISS learning process is recorded. There are two foci during the learning process: firstly the know-how of the programming language and structures of the OpenCMISS Iron environment, and secondly the continuum mechanics principles underpinning the coding. The first focus is recorded in the section titled User Remarks, where any tips or know-hows of coding practice are recorded as they are learnt. The second focus is laid out in the section titled Examples where reality checks on toy problems are explained and the results of simulation displayed.

# User remarks

## Important variable names

DZDNU Deformation Gradient Tensor

DZDNUT

Jznu

AZL Covariant right Cauchy Deformation Tension

AZL\_SQUARED= C2

AZU Contravariant Deformation Tensor

E Green-Lagrange strain tensor

PIOLA\_TENSOR The Second Piola Kirchhoff Tensor T

P Hydrostatic pressure

C Coefficients of the isotropic constitutive model

## Finite Element method

Equilibrium equation gives

Residual equation

Where for the Galerkin method

For finite elasticity, the discretised equation will be non-linear

To solve this we will use the Newton-Raphson method

Where

And

## Code Notes

**Coordinate System**

Source code: Coordinate\_routines.f90

* CreateStart  
  Default dimensions 3, focus (for curvilinear coordinates) is 1, origin at 0,0,0.

A pointer to the coordinate system object is created.

* CreateFinish  
  Write out diagnostic messages if option is set (diagnostics 1)  
  Diagnostic message shows the number of coordinate systems, and for each coordinate system shows the user number, type and dimensions.   
  NB: Only on type of coordinate system is currently implemented –the rectangular Cartesian.

**Region**

Source code: region\_routines.f90

* CreateStart  
  Find user number  
  Make sure the user number sin’t already associated  
  Make sure parent region has been finished.
  + Region initialise  
    Initialize the coordinate system, data points, nodes, meshes… everything that is enveloped in the region.

Set user number, label and coordinate system of the region, which is inherited from the parent region.

Update number of subregions.

**Basis**

Source code: basis\_routines.f90

* CreateStart  
  Check that user number hasn’t already been used.
  + Basis initialise  
    Initalises the attributes of basis such as: user number, type, number of xi, number of nodes…, number of collapsed xi.

Set type, number of basis functions, number of xi directions (default 3), default trilinear lagrange, default not collapsed for all 3 xi directions, initialise quadrature.

* CreateFinish  
  If using lagrange hermite tensor product type then:
  + Basis LHTP family create  
    Create parent basis
    - Basis LHTP basis create  
      Set the number of xi, number of partial derivatives which is equal to the (number of xi)^2 + 2  
      According to the interpolation scheme, set the number of nodes.  
      If a xi direction is collapsed then set degenerate boolean to true and record which xi is collapsed.

# Useful expressions

Principle of linear momentum

Principle of angular momentum

Energy balance

Constitutive Representation

Incompressibility

Incompressible Constitutive Relation

Piola Stresses

1st Piola Kirchhoff Stress describes the stress vector in deformed state measured per area of corresponding surface in un-deformed state.

First we describe the deformed surface in terms of the un-deformed surface:

Multiply both sides by , we get:

Since

Now multiply both sides with respect to

Hence, the stress tensors can be translated.

Hence the 1st Piola Kirchhoff stress

The 2nd Piola Kirchhoff stress describes the stress vector in the un-deformed state measured per area of corresponding surface in un-deformed state.

The un-deformed stress is related to the deformed stress tensor by the same way that a material vector dX is related by deformation to the spatial vector dx, i.e. related through the inverse of the deformation gradient tensor F.

Tensor identities

Invariant Derivatives

# Examples

The following simple example problems were implemented in OpenCMISS and a reality check was done for each. These will be the building blocks for constructing the left ventricular model mechanics. The examples will begin with a simple unit cube on which various mechanics a tested. Then the geometry will become more complex, taking the form of a cylinder, and then eventually that of the 16 element left ventricular model.

## Uniaxial Extension

Using legacy CMISS, a uniaxial extension in the x direction of 10% was implemented (CMISS examples). The deformation can be described by:

Where

Since the material is incompressible,

Hence the deformation gradient tensor, and right deformation tensor are:

The Mooney-Rivlin constitutive model is used. Therefore the 2nd Piola Kirchhoff stress in terms of the C tensor is stated as:

The hydrostatic pressure was listed in CMISS as -13.37685, we need to multiply this by a factor of two to find the value for p in the previous equation. Hence:

We then convert the 2nd Piola Kirchhoff stress to the Cauchy stress tensor using the transformation:

The value of the Cauchy stress and the 2nd Piola stress matches the values given by CMISS.

We could also apply the stress vector through a force application on the four nodes of the face 2,4,6,8. The amount of force applied on this surface is calculated:

Which means that the force applied on each node is:

This force should give us the deformation described before (strain of 10% in the x direction).

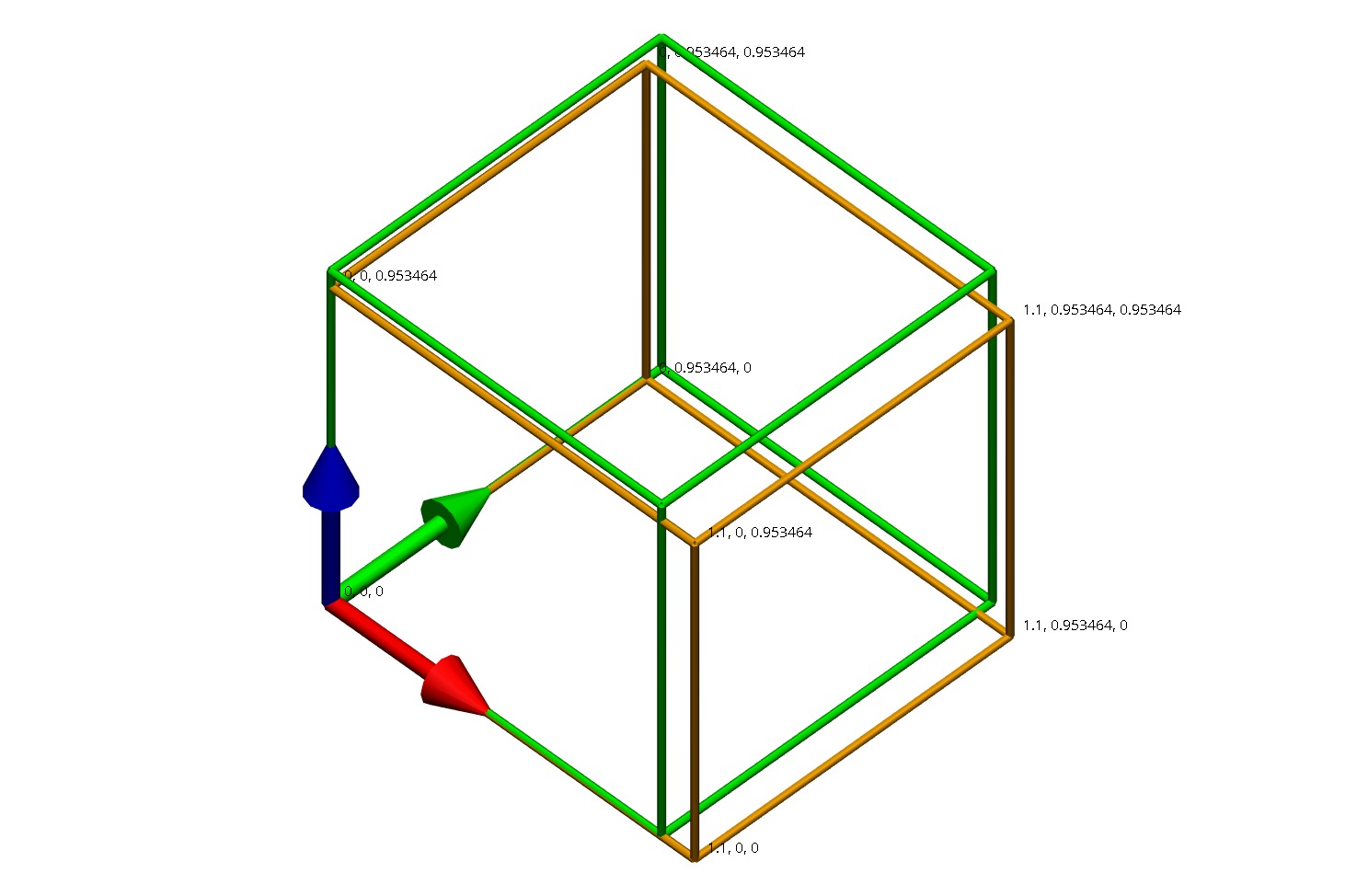
This was implemented in OpenCMISS by applying the force on each of the four nodes as a flux term. The following deformed state was produced. The strains were as expected.

Figure Deformed cube (gold skeleton) and un-deformed cube (green skeleton) with deformed coordinates of each node labelled.

Python script: UniAxialNodalForces.py

## Uniaxial Pressure

To check whether OpenCMISS applies pressure in the same was as we would expect, a pressure boundary condition toy problem was implemented. We wish to reproduce the 10% extension as in the previous example; hence the stress we wish to apply on the face has a value of:

This stress is the force of deformed state measured with respect to the deformed surface. It seems, through experimentation, that OpenCMISS takes the Cauchy stress value as pressure input and so no conversion to 1st PK stress was necessary.

The deformations were identical to Figure 1.

Python code: UniAxialPressure.py

## Simple Shear

Extra toy problem was built in OpenCMISS to compare with legacy CMISS of shear of unit cube in xz-plane.

The deformed cube is shown below.

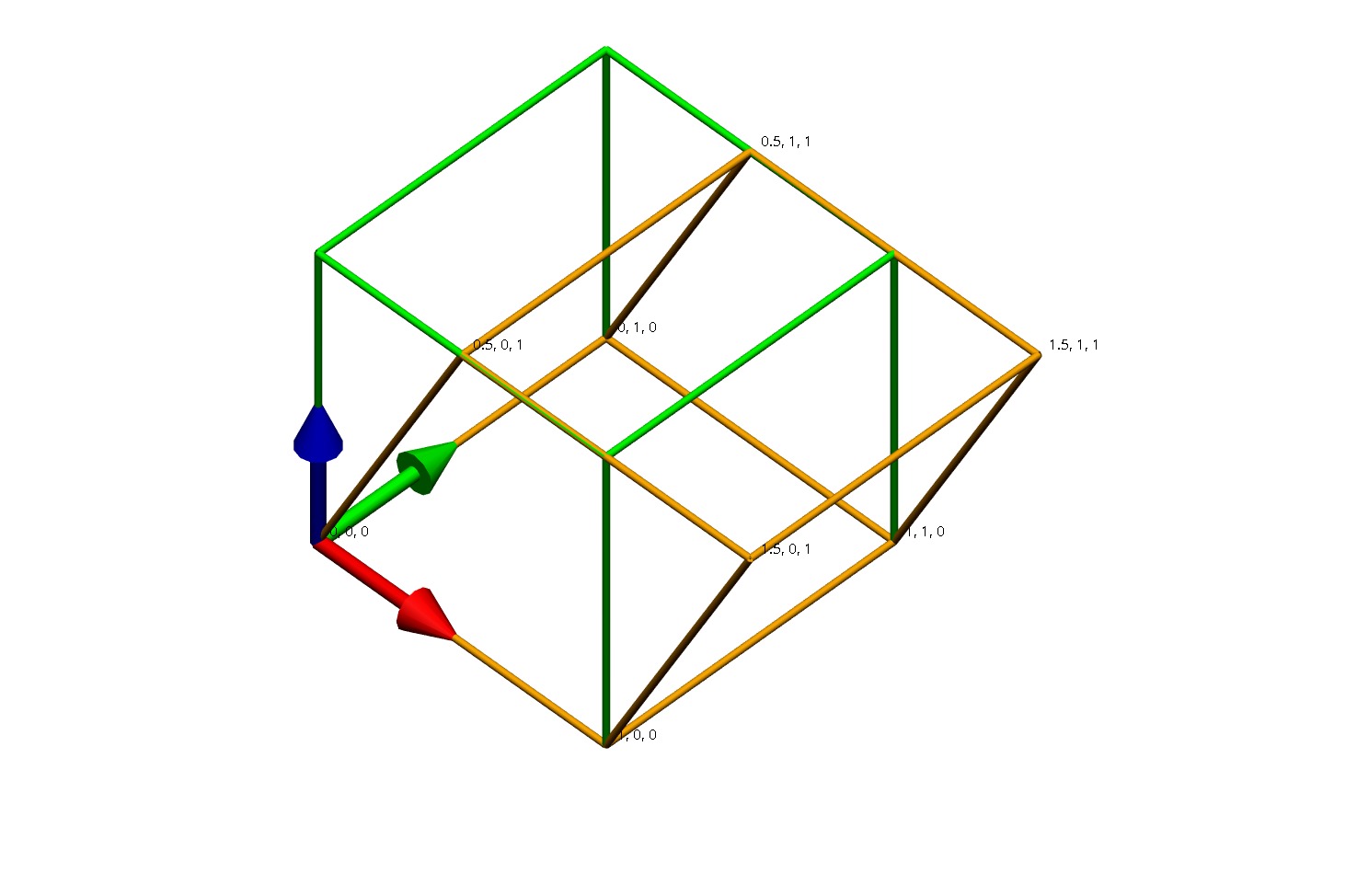


Figure Shear in xz plane of unit cube

The deformation in this simple shear can be described as

And the deformation gradient tensor is therefore

And

Hence the 2nd PK stress is

Where C1 is 1.0 and C2 is 0.2. The hydrostatic pressure was given by both CMISS and OpenCMISS as 1.45 (3 s.f.)

This matches with the CMISS stresses.

To replicate this deformation by applying nodal forces, we transform the stress into the 1st Piola Kirchhoff stress (since this is measured with respect to the un-deformed areas):

Python script: SimpleShear.py

## Transversely Isotropic

The Guccione transversely isotropic constitutive model is used. We wish to apply nodal forces again to create a 10% extension in the x direction. Therefore the deformation gradient tensor and the right deformation tensor are the same as the first example:

Since the Guccione constitutive model is written in terms of the Green strain tensor, therefore it is calculated:

The strain energy function is given as:

Hence the derived second Piola Kirchhoff stress tensor is:

Using the Green strain tensor stated before, and computing the inverse of the right deformation tensor, the stress reduces to:

The coefficients are given as:

Hence

We can then deduce that the hydrostatic pressure p is 0.1716, giving

The Cauchy stress tensor is therefore

This is applied as a pressure on the face, and the deformation is shown below. There was a good match with the deformation we expected.

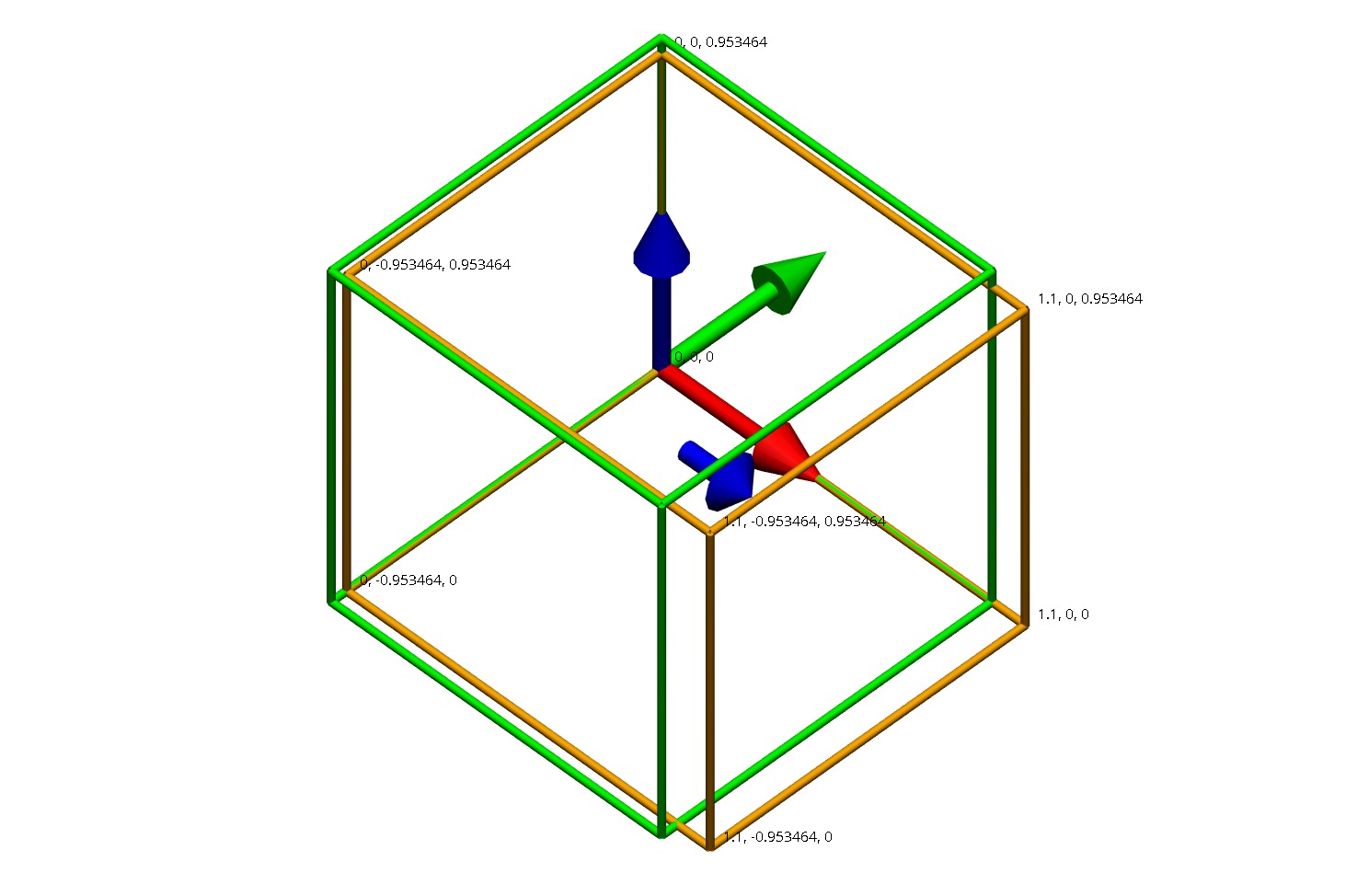


Figure Deformation of transversely isotropic unit cube.

Next we rotate the fibre with and still with to reproduce a uniaxial extension in the x direction. In Martyn’s thesis, chapter 4.3.2, it is written that fibres lie in the xi1 and xi2 plane and the fibre angle is defined with respect to the xi1 coordinate. The angles listed above correspond to the fibre angle (, the imbrication angle ( and the sheet angle which can be used to transform components of the material vectors between the microstructural coordinate system to the global coordinate system (or wall coordinate system in terms of the heart). This transformation consists of three coordinate rotations.

Let the base vectors of the material microstructural coordinate system be (a,b,c), and the base vectors of the cube coordinate system be (f,g,h). This is the coordinate system in which the Guccione material law is expressed. The first rotation is around the fibre

|  |  |
| --- | --- |
| Rotation | Coordinate system |
| angle about fibre (a) axis |  |
| angle about intermediate imbrication () axis |  |
| angle about final sheet (h) axis |  |

The first rotation is performed by

The second rotation is performed by

And the final rotation is performed by

And so, the entire transformation from the material coordinates to the global coordinates is

And the reverse of that is

Therefore if we wish for a uniaxial extension in the x direction to occur in the global coordinate system, we would need to transform the deformation gradient tensor into the material coordinate system before we can apply the constitutive law. After we have obtained the stress tensors we then need to transform them back into global coordinate systems in order to apply the appropriate loading conditions.

To transform a tensor however, we need to:

The deformation gradient tensor is not the same as the previous example since the 10% extension is now in a cross-fibre direction. To get this tensor we first solved a uniaxial extension problem prescribing the displacement.

Hence

Transform back to global coordinate system.

Convert to Cauchy stress

Python script: UniAxialTransverselyIsotropicTriLinear.py

## Transversely Isotropic in CellML

With an un-rotate fibre orientation another model was built in which CellML was used to describe the constitutive model.

Python script: TransverselyIsotropicCellML.py

## Active Contraction

The active contraction stress is added on to the 2nd PK stress tensor. This can be implemented in CellML.