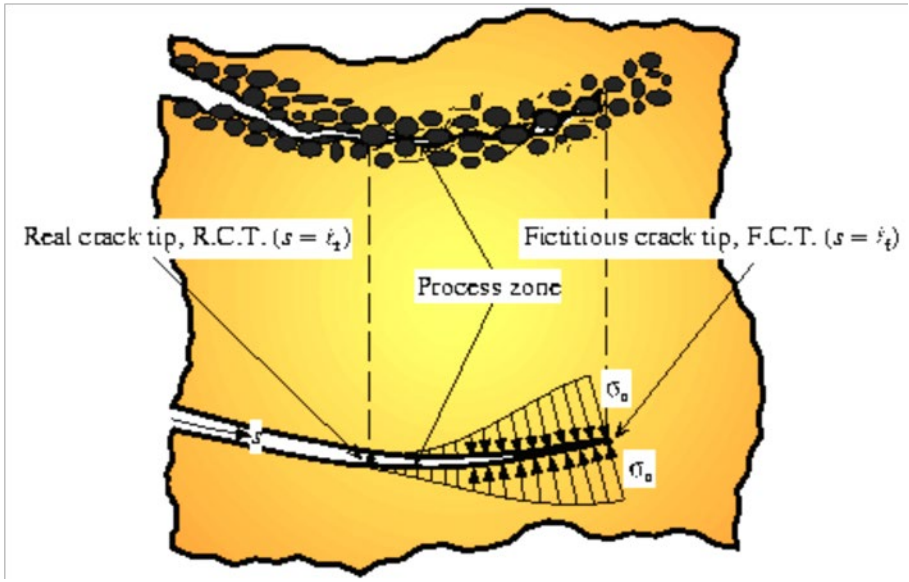
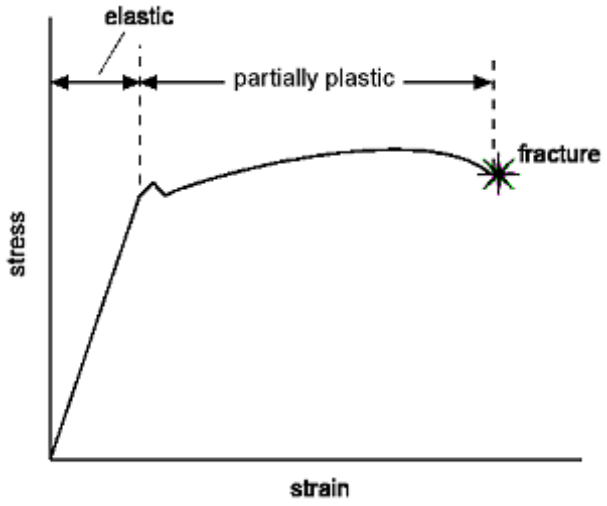
Constitutive Modelling in Kratos

October 2013

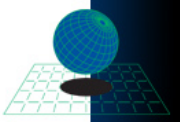




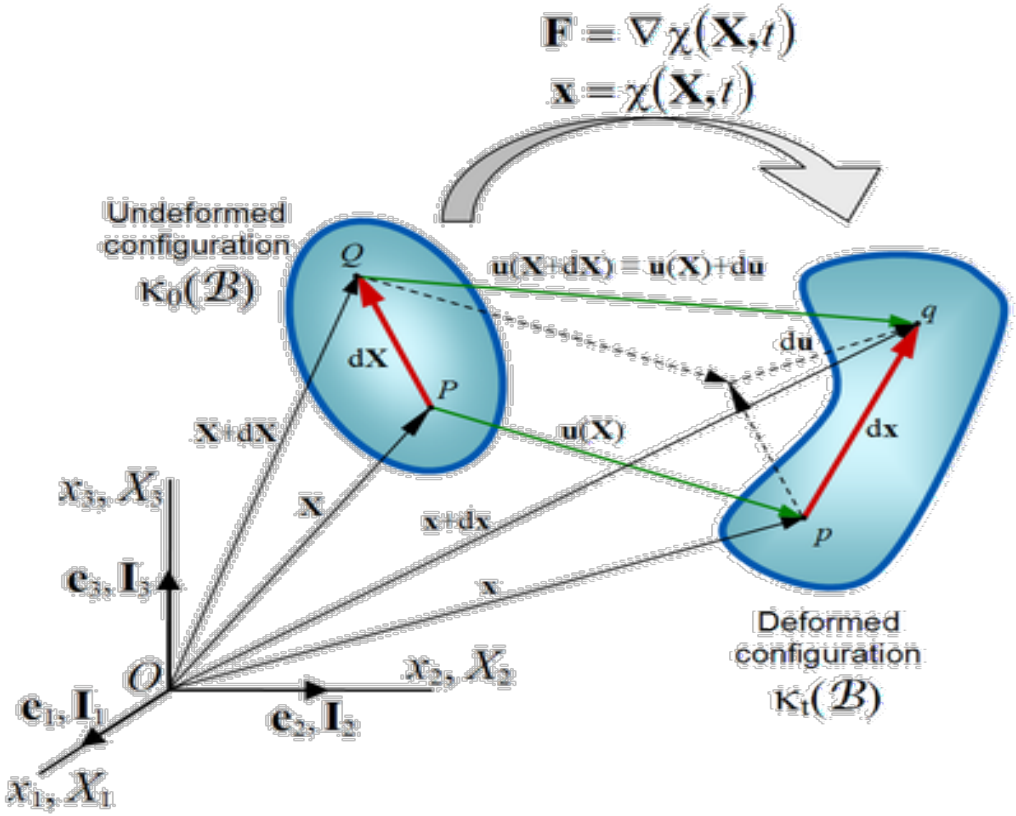
kratos/kratos/includes/constitutive_law.h

Outline

- Note on kinematics of the continuum
- Definition of the Strain and Stress measures
- Constitutive Law interface and methods
- Constitutive Law structure in object programming
- Examples of operation



Kinematics of the continuum



Deformation

- Deformation Gradient
- Left Cauchy Green
- Right Cauchy Green

Finite Strains

- Green-Lagrange
- Almansi
- Henky

Infinitesimal Strains

- Infinitesimal Strain

Constitutive Law Strain Measures

enum **StrainMeasure** { //deformation and strain measures:

1. **StrainMeasure_Infinitesimal,** $\varepsilon = \frac{1}{2}(\frac{d\mathbf{u}_i}{d\mathbf{x}_j} + \frac{d\mathbf{u}_j}{d\mathbf{x}_i})$
2. **StrainMeasure_GreenLagrange,** $\mathbf{E} = \frac{1}{2}(\mathbf{F}^{\mathbf{T}}\mathbf{F} - \mathbf{1}) = \frac{1}{2}(\mathbf{C} - \mathbf{1})$
3. **StrainMeasure_Almansi,** $\mathbf{e} = \frac{1}{2}(\mathbf{1} - \mathbf{F}^{-\mathbf{T}}\mathbf{F}^{-\mathbf{1}}) = \frac{1}{2}(\mathbf{1} - \mathbf{b}^{-\mathbf{1}})$
4. **StrainMeasure_Hencky_Material,** $\mathbf{H} = \ln \mathbf{U}$
5. **StrainMeasure_Hencky_Spatial,** $\mathbf{h} = \ln \mathbf{v}$
6. **StrainMeasure_Deformation_Gradient,** $\mathbf{F} = \frac{d\mathbf{x}}{d\mathbf{X}}$
7. **StrainMeasure_Right_CauchyGreen,** $\mathbf{C} = \mathbf{F}^{\mathbf{T}}\mathbf{F}$
8. **StrainMeasure_Left_CauchyGreen** $\mathbf{b} = \mathbf{F}\mathbf{F}^{\mathbf{T}}$

};

$\mathbf{F} = \mathbf{R}\mathbf{U} \qquad \mathbf{F} = \mathbf{v}\mathbf{R}$

$\mathbf{U} = \sqrt{\mathbf{C}}$

$\mathbf{v} = \mathbf{R}\mathbf{U}\mathbf{R}^{\mathbf{T}}$

Constitutive Law Stress Measures

enum **StressMeasure** { //stress measures:

1. **StressMeasure_PK1,** $\mathbf{P} = J\boldsymbol{\sigma}\mathbf{F}^{-\mathbf{T}}$ $\mathbf{P} = \mathbf{FS}$
2. **StressMeasure_PK2,** $\mathbf{S} = J\mathbf{F}^{-1}\boldsymbol{\sigma}\mathbf{F}^{-\mathbf{T}}$
3. **StressMeasure_Kirchhoff,** $\boldsymbol{\tau} = J\boldsymbol{\sigma}$ $\boldsymbol{\tau} = \mathbf{FSF}^{\mathbf{T}}$
4. **StressMeasure_Cauchy,** $\boldsymbol{\sigma} = \mathbf{c}\ \boldsymbol{\varepsilon}$

};

Material Response Models

Stress Measure $\mathbf{S} = \mathbf{C}\ \mathbf{E}$ **Strain Measure**

Constitutive Tensor

Constitutive Law interface and methods

General Method to get the material response for a specific "Stress Measure":

CalculateMaterialResponse(*Parameters*, *Stress Measure* ***);***

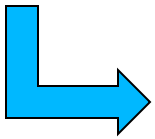
Specific Methods for the requested stress measures:

CalculateMaterialResponsePK1 *(Parameters)*

CalculateMaterialResponsePK2 *(Parameters)*

CalculateMaterialResponseKirchhoff *(Parameters)*

CalculateMaterialResponseCauchy *(Parameters)*



StressVector

ConstitutiveMatrix

Constitutive Law interface and methods

**** KINEMATIC* PARAMETERS:***

'''mDeterminantF''' *opy of the determinant of the Current Deformation Gradient (although Current F is also included as a matrix) (input data)*

'''mDeterminantF0''' *copy of the determinant of the Total Deformation Gradient (although Total F0 is also included as a matrix) (input data)*

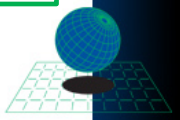
'''mpDeformationGradientF''' *pointer to the current deformation gradient (can be an empty matrix if a linear strain measure is used) (input data)*

'''mpDeformationGradientF0''' *pointer to the total deformation gradient (can be an empty matrix if a linear strain measure is used) (input data)*

'''mpStrainVector''' *pointer to the current strains (total strains) (input data) (can be also an output with COMPUTE_STRAIN flag)*

'''mpStressVector''' *pointer to the current stresses (output with COMPUTE_STRESS flag)*

'''mpConstitutiveMatrix''' *pointer to the material tangent matrix (output with COMPUTE_CONSTITUTIVE_TENSOR flag)*



Constitutive Law interface and methods

**** GEOMETRIC* PARAMETERS:***

'''mpShapeFunctionsValues''' *pointer to the shape functions values in the current integration point (input data)*

'''mpShapeFunctionsDerivatives''' *pointer to the shape functions derivatives values in the current integration point (input data)*

'''mpElementGeometry''' *pointer to the element's geometry (input data)*

**** MATERIAL PROPERTIES* PARAMETERS:***

'''mpMaterialProperties''' *pointer to the material Properties object (input data)*

**** PROCESS PROPERTIES* PARAMETERS:***

'''mpCurrentProcessInfo''' *pointer to current ProcessInfo instance (input data)*

Constitutive Law interface and methods

**** FLAGS * PARAMETERS:***

'''mOptions''' *flags for the current Constitutive Law Parameters (input data)*

COMPUTE_STRAIN

COMPUTE_STRESS

COMPUTE_CONSTITUTIVE_TENSOR

ISOCHORIC_TENSOR_ONLY

VOLUMETRIC_TENSOR_ONLY

TOTAL_TENSOR

INITIAL_CONFIGURATION

LAST_KNOWN_CONFIGURATION

FINAL_CONFIGURATION

FINALIZE_MATERIAL_RESPONSE

Constitutive Law interface and methods

CHECK

Get the constitutive law features and check if is appropriate

"***Features***" in order to get the constitutive law characteristics*

* its variables will be used to check constitutive law and element compatibility

**** mOptions*** flags with the current constitutive law characteristics

**** mStrainSize*** double with the strain vector size

**** mStrainMeasures*** vector with the strain measures accepted by the constitutive law

- **FINITE_STRAINS**
- **INFINITESIMAL_STRAINS**

- **THREE_DIMENSIONAL_LAW**
- **PLANE_STRAIN_LAW**
- **PLANE_STRESS_LAW**
- **AXISYMMETRIC_LAW**

- **U_P_LAW**
- **ISOTROPIC**
- **ANISOTROPIC**

Constitutive Law interface and methods

STARTING

Set initial constitutive law variables

InitializeMaterial (MaterialProperties, ElementGeometry, ShapeFunctionsValues);

InitializeSolutionStep (MaterialProperties, ElementGeometry, ShapeFunctionsValues, CurrentProcessInfo);

InitializeNonLinearIteration (MaterialProperties, ElementGeometry, ShapeFunctionsValues, CurrentProcessInfo);

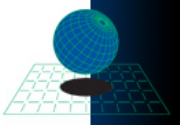
Constitutive Law interface and methods

ENDING

Update constitutive law variables

FinalizeSolutionStep (MaterialProperties, ElementGeometry, ShapeFunctionsValues, CurrentProcessInfo);

FinalizeNonLinearIteration (MaterialProperties, ElementGeometry, ShapeFunctionsValues, CurrentProcessInfo);



Constitutive Law interface and methods

ENDING

Update constitutive law variables, if a material response computation is needed.

General Method to Finalize the material response for a specific "Stress Measure":

FinalizeMaterialResponse(*Parameters*, *Stress Measure* ***);***

Specific Methods for the requested stress measures:

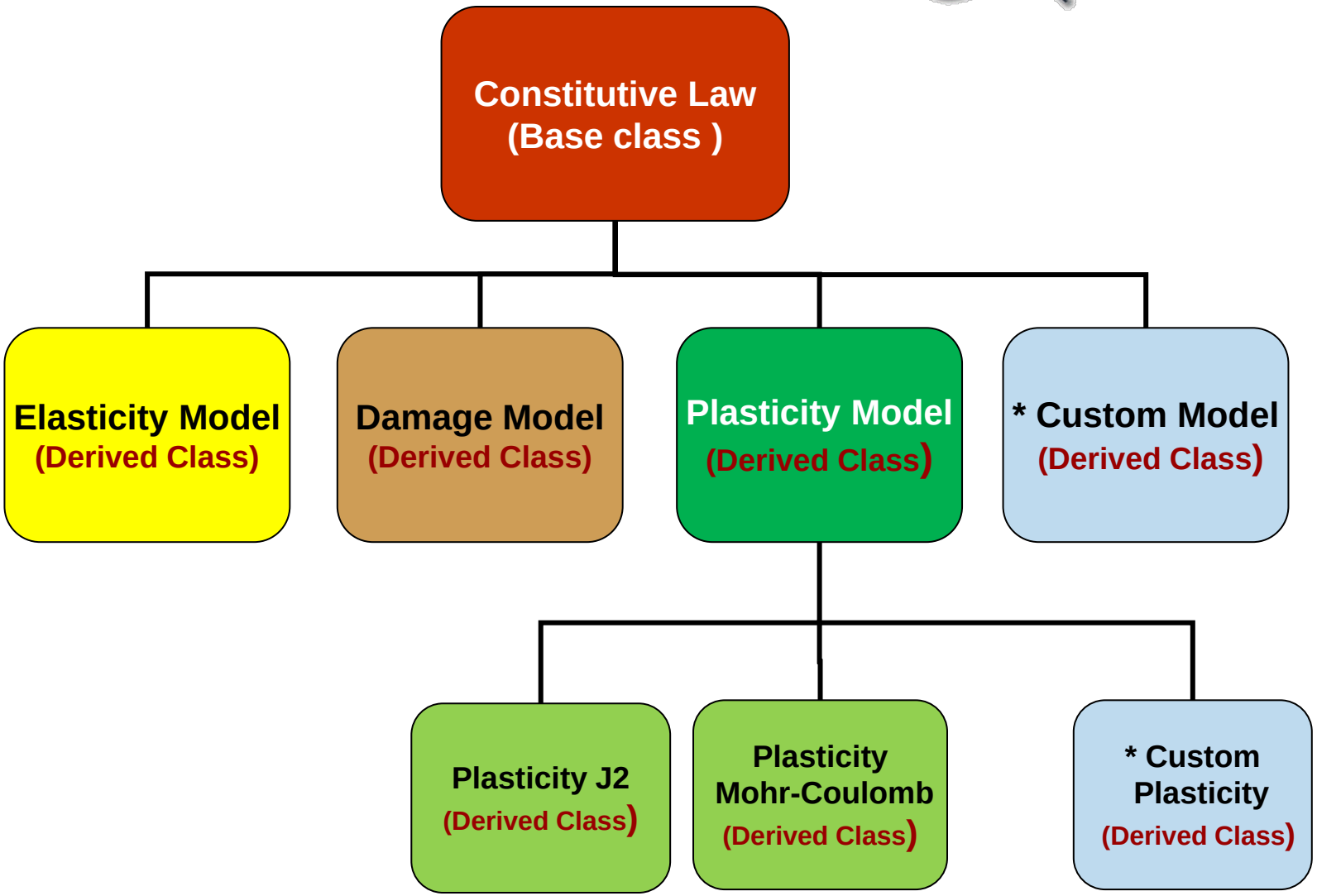
FinalizeMaterialResponsePK1 *(Parameters)*

FinalizeMaterialResponsePK2 *(Parameters)*

FinalizeMaterialResponseKirchhoff *(Parameters)*

FinalizeMaterialResponseCauchy *(Parameters)*

Constitutive Law Structure



Constitutive Law Structure

***NEW PLASTICITY MODEL**

Behaviour can be described with the definition of the :

Input Data (defined in the constructor)

FLOW RULE

Radial Return definition

YIELD CRITERION

Yield Surface state function

HARDENING LAW

Hardening or softening function

Constitutive Law Structure

- Each constitutive Law must include the Stress Calculation and the Constitutive Tensor Calculation

- Constitutive laws do not store elastic or kinematic variables but they store and manage internal variables
  - Internal variables are “Member Variables”
  - The update of the internal variables is done by a class method

  (i.e. In plasticity internal variables belong to the Flow Rule)

- The Constitutive Law “Parameters” structure, is only an interface between the integration point variables and the constitutive law calculation methods.

  It only contains pointers to the integration point variables. Parameters have to be set and get by means of their Set and Get methods.

  (i.e. SetDeformationGradientF(F) or Matrix& F = GetDeformationGradientF())

Examples of operation

- Create a constitutive law from Properties:

1) The variable name gives direct access to the property

   double ElasticModulus = Properties [YOUNG_MODULUS]

   double Density = Properties [DENSITY]

2) The constitutive law can be initialized from properties

   a) Properties MyProperties = GetProperties(); →(i. e. in elements, set from cloning pointer)

   - ConstitutiveLawPointer MyConstitutiveLaw = Properties [CONSTITUTIVE_LAW] ->Clone();

   b) Directly creating the constitutive law

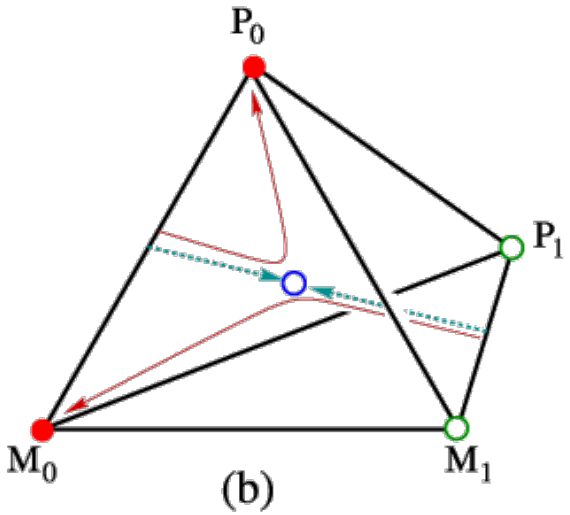
   - ConstitutiveLawPointer MyConstitutiveLaw = new LinearElastic3DLaw();
   - LinearElastic3DLaw MyConstitutiveLaw;

Examples of operation

- Element Variables : GEOMETRY

  MATERIAL PROPERTIES

  CONSTITUTIVE LAW VECTOR



Size of vector = number of integration points

INTEGRATION POINT LOCAL VARIABLES:

-> StrainVector $\varepsilon = \frac{1}{2}(\frac{d\mathbf{u}_i}{d\mathbf{x}_j} + \frac{d\mathbf{u}_j}{d\mathbf{x}_i})$

(computed from nodal displacements)

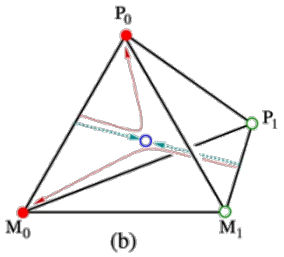
-> StressVector (initialize)

-> ConstitutiveMatrix (initialize)

-> ShapeFunctions (get from geometry)

....

Examples of operation



- **ELEMENT CALCULATION** on integration point "*i*":

```
-- Create Constitutive Law Parameters
ConstitutiveLaw::Parameters   Values(  Geometry,  GetProperties,  CurrentProcessInfo );

-- Set Constitutive Law Options
Flags & ConstitutiveLawOptions = Values.GetOptions();
ConstitutiveLawOptions.Set   (ConstitutiveLaw::COMPUTE_STRESS);
ConstitutiveLawOptions.Set   (ConstitutiveLaw::COMPUTE_CONSTITUTIVE_MATRIX);

Loop on " i " :

    -- Set Constitutive Law Variables
    Values.SetStrainVector          (StrainVector);
    Values.SetStressVector          (StressVector);
    Values.SetConstitutiveMatrix  (ConstitutiveMatrix);

    Values.SetShapeFunctionsDerivatives   (DerivativesShapeFunctions);
    Values.SetShapeFunctionsValues          (ShapeFunctions)
    ....
    ConstitutiveLawVector [ i ] -> CalculateMaterialResponseCauchy( Values )
    ....
```

"Calculate System Matrices"

Examples of operation

- **CONSTITUTIVE LAW CALCULATION** : (i.e. LinearElastic3DLaw())

```
void CalculateMaterialResponseCauchy ( Paramaters& Values )
    {
      -- Get Values to compute the constitutive law:
      Flags &Options                          = Values.GetOptions();

      Vector& StrainVector                    = Values.GetStrainVector();
      Vector& StressVector                    = Values.GetStressVector();
      Matrix& ConstitutiveMatrix              = Values.GetConstitutiveMatrix();
      const Properties& MaterialProperties    = Values.GetMaterialProperties();

     -- Get Properties
      const double& YoungModulus              = MaterialProperties [ YOUNG_MODULUS ];
      const double& PoissonCoefficient        = MaterialProperties [ POISSON_RATIO ];

      -- Compute Stress and Constitutive Matrix
      if ( Options.Is( ConstitutiveLaw::COMPUTE_STRESS ) ) {
        CalculateLinearElasticMatrix  ( ConstitutiveMatrix, YoungModulus, PoissonCoefficient );
        CalculateStress  ( StrainVector, ConstitutiveMatrix, StressVector )
      }
    }
```

Examples of operation

Methods GetValue(…) and SetValue(…)

They set and return the value of a custom variable.

```
double& GetValue( const Variable<double>& rThisVariable, double& rValue )
  {
    if ( rThisVariable == DAMAGE )
    {
      mpFluencyCriteria->GetValue(rThisVariable, rValue);
    }
    if ( rThisVariable == YIELD_STRESS)
    {
      mpFluencyCriteria->GetValue(rThisVariable, rValue);
    }
    return rValue;
  }

void SetValue( const Variable<double>& rThisVariable, double& rValue )
  {
    if ( rThisVariable == DAMAGE )
    {
      mpFluencyCriteria->SetValue(rThisVariable, rValue);
    }
    if ( rThisVariable == YIELD_STRESS)
    {
      mpFluencyCriteria->SetValue(rThisVariable, rValue);
    }
  }
```

Examples of operation

Methods TransformStrains(…) and TransformStresses(…)

They perform the covariant or contravariant transformation to transform most of the StrainMeasures and the StressMeasures.

```
Vector& TransformStrains (Vector& rStrainVector,
                          const Matrix &rF,
                          StrainMeasure rStrainInitial,
                          StrainMeasure rStrainFinal)
```

```
Matrix& TransformStresses (Vector& rStressVector,
                           const Matrix &rF,
                           const double &rdetF,
                           StressMeasure rStressInitial,
                           StressMeasure rStressFinal)
```

Examples of operation

Notation note:

msIndexVoigt3D6C [6][2] =
{ {0, 0}, {1, 1}, {2, 2}, {0, 1}, {1, 2}, {0, 2} };

msIndexVoigt2D4C [4][2] =
{ {0, 0}, {1, 1}, {2, 2}, {0, 1} };

msIndexVoigt2D3C [3][2] =
{ {0, 0}, {1, 1}, {0, 1} };

$$\boldsymbol{\sigma}_{ij} = \boldsymbol{C}_{ijkl}\boldsymbol{\epsilon}_{kl}$$

Voigt notation

$$\begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{12} \end{Bmatrix} = \begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{xy} \end{Bmatrix}$$

$$\begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \varepsilon_{12} \end{Bmatrix} = \begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{xy} \end{Bmatrix}$$

$$\begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \varepsilon_{12} \\ \varepsilon_{23} \\ \varepsilon_{13} \end{Bmatrix} = \begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{xy} \\ \varepsilon_{yz} \\ \varepsilon_{xz} \end{Bmatrix}$$

Examples of operation

Reading constitutive law (python)

- Create a Law from material.py file

```
Properties[2].SetValue(DENSITY,2700);
Properties[2].SetValue(POISSON_RATIO, 0.25);
Properties[2].SetValue(YOUNG_MODULUS, 50E9);
Properties[2].SetValue(THICKNESS, 1.00);
ConstitutiveLaw2   = LinearElasticPlaneStress2DLaw ( )
Properties[2].SetValue ( CONSTITUTIVE_LAW, ConstitutiveLaw2 )
```

- Import structural Aplication

```
from KratosMultiphysics import *
from KratosMultiphysics.SolidMechanicsApplication import *
```

Examples of operation

hyperelastic_3D_law.cpp
hyperelastic_plane_strain_2D_law.cpp
hyperelastic_axisym_2D_law.cpp

hyperelastic_U_P_3D_law.cpp
hyperelastic_U_P_axisym_2D_law.cpp
hyperelastic_U_P_plane_strain_2D_law.cpp
hyperelastic_plastic_U_P_3D_law.cpp

hyperelastic_plastic_3D_law.cpp
hyperelastic_plastic_plane_strain_2D_law.cpp

hyperelastic_plastic_J2_3D_law.cpp
hyperelastic_plastic_J2_plane_strain_2D_law.cpp

linear_elastic_3D_law.cpp
linear_elastic_axisym_2D_law.cpp
linear_elastic_plane_strain_2D_law.cpp
linear_elastic_plane_stress_2D_law.cpp

/kratos
/applications
/SolidMechanicsApplication
/custom_constitutive
/

THANK YOU FOR YOUR ATTENTION !

/// Any questions ? ///