PyMecha Documentation

Release v1.0

Boris Burgarella

September 02, 2014

CONTENTS

This is the manual created for the PyMecha software by Boris Burgarella, LMA, CNRS. For any questions or comments, send an email to: boris.burgarella@lma.cnrs-mrs.fr . Contents: Pymecha was written by Boris Burgarella the objective of this module is to simplify the micro-mechanics calculations using python

CHAPTER

ONE

MATHEMATIC FUNCTIONS

Pymecha.**importTensorMathematica**(*fileName*)

This function import a tensor exported from mathematica in a text file. argument: filename, string containing the filename and it's extension

returns: Table representing the tensor

Pymecha.**testdelabase**(*El*, *Jt*, *F*, *tF*, *Kt*, *Kl*)

This function is made to test that the base used for projecting the tensors is right.

Arguments: 3*3*3*3 Tensors El, Jt, F,tF,Kt,Kl Returns: Resultat, list of which length if 9. 1 means the equation is verified 0 means it isn't.

If you want to use a base, remember that the whole Resultat list must be at 1 for the base to be usable.

Pymecha.**delta**(*i*, *j*)

Kroeneker symbol. arguments: i and j type of arguments: int i and int j Returns: 1 if i == j and 0 elsewhat

Pymecha.**matriceordreN**(*n*)

takes int n as input, and returns a 3*3*3*3*3 [...] n times tensor.

example: matrice(2) = [0,0,0] [0,0,0] [0,0,0]

the returned tensor is full of 0

Pymecha.**passage24**(*a*)

this function is written to help the function that transform the 6x6 stiffness matrix back to it's normal form 3*3*3*3 arguments: int a

returns: int values: [i,j] vector: if a <= 2: i=j=a if a = 3: i = 1 and j = 2 if a = 4: i = 0 and j = 2 if a = 5: i = 0 and j = 1 This function is the inverse fuction of passage42(i,j)

Pymecha.**passage42**(*i*, *j*)

this function is written to help the function that transform the 6x6 stiffness matrix back to it's normal form 3*3*3*3 arguments: int [i,j] returns: int valuea: [i,j] vector: if i=j: a = i = j if i = 1 and j = 2: a = 3 if i = 0 and j = 2: a = 4 if i = 0 and j = 1: a = 5 This function is the inverse fuction of passage24(a)

Pymecha.**checkequal0**(*A*)

Booléan function that takes a tensor A (3x3x3x3) as entry and returns True if A = 0 and False is A isn't null

Pymecha.**checkequal**(*A*, *B*)

Booléan function that takes two 6x6 matrices A and B as entry and returns True if A = B and False if not

Pymecha.**NumSup**(*A*)

Basic function that returns the upper value of a list (float of int)

Pymecha.**NumInf**(*A*)

Basic function that returns the lower value of a list (float of int)

CHAPTER

TWO

2ND ORDER TENSOR CALCULATION FUNCTIONS

Pymecha.**produittensoriel12**(*A*, *B*)

takes two vectors A and B (dimensions: 3x1) and returns the tensorial product of A and B

Type: int or float returns: Matrix of float, dimension NxN

Functions called by this one: matriceordreN(n)

Pymecha.**soustrMatordre2**(*A*, *B*)

I wrote this function when I wasn't using numpy, and I needed a function to substrat two matrix, it takes two 3x3 matrix as arguments and returns a 3x3 matrix equals to A-B

Functions called by this one: matriceordreN(n)

Pymecha.**addMat2**(*I*, *J*)

I wrote this function when I wasn't using numpy, and I needed a function to add two matrix, it takes two 3x3 matrix as arguments and returns a 3x3 matrix equals to A+B

Functions called by this one: matriceordreN(n)

CHAPTER

THREE

4TH ORDER TENSOR CALCULATION FUNCTIONS

Pymecha.**transpose4**(*mat*)

Takes a 3x3x3x3 tensor as arguments

returns the transposed tensor

Functions called by this one: matriceordreN(n)

Pymecha.**definitionI**()

this function determines the identity tensor for the symmetrical 4rth order tensor vectorial space arguments: null returns: 3x3x3x3 tensor I2

Functions called by this one: matriceordreN(n)

Pymecha.**produittensoriel24**(*A*, *B*)

takes two vectors A and B (dimensions: 3x3) and returns the tensorial product of A and B

Type: int or float returns: Matrix of float, dimension 3x3x3x3

Functions called by this one: matriceordreN(n)

Pymecha.**soustrMat**(*I*, *J*)

I wrote this function when I wasn't using numpy, and I needed a function to substrat two matrix, it takes two 3x3x3x3 matrix as arguments and returns a 3x3x3x3 matrix equals to A-B

Functions called by this one: matriceordreN(n)

Pymecha.**addMat**(*I*, *J*)

I wrote this function when I wasn't using numpy, and I needed a function to add two matrix, it takes two 3x3x3x3 matrix as arguments and returns a 3x3x3x3 matrix equals to A+B

Functions called by this one: matriceordreN(n)

Pymecha.**produitMatScal**(*Scal*, *Mat*)

simple product between a scalar and a matrix of dimension 3x3x3x3 type: float or int returns: Matrix of float, dimention 3x3x3x3

Pymecha.**produitContracte42**(*A*, *a*)

This functions calculates the "contracted product" (name directly translated from french as I don't know the english name. this product is defined by this equations: Aa[i][j] = A[i][i][k][l]*a[k][l] (Einstein notation)

arguments: a matrix of dimension 3x3 and A tensor of dimension 3x3x3x3 returns Aa matrix of dimension 3x3

Pymecha.**produitContracte44**(*A*, *B*)

This function calculates the "contracted product" (name directly translated from french as I don't know the english name. this product is defined by this equations: AB[i][j][k][l] = A[i][j][m][n]*B[m][n][k][l] (Einstein notation)

arguments: a matrix of dimension 3x3x3x3 and A tensor of dimension 3x3x3x3 returns AB matrix of dimension 3x3x3x3

CHAPTER

FOUR

NOTATION FUNCTIONS

Pymecha.**Voigt**(*matrice4*)

This function takes a 4rth order symmetrical tensor and returns it in Voigt notation it is the inverse function of UnVoigt(matrix) Arguments: 4rth order tensor of dimension 3x3x3x3 returns: 6x6 matrix

Functions called by this one: passage24(a)

Pymecha.**UnVoigt**(*matrice2*)

This function takes a 6x6 matrix and returns it's equivalent 4rth order symmetrical tensor it is the inverse function of Voigt(Tensor) Arguments: 6x6 matrix Returns: 4rth order tensor of dimension 3x3x3x3

Functions called by this one: passage42(i,j)

Pymecha.**SixXSix**(*Matrice4*)

This function takes a 4rth order symmetrical tensor and returns it in modified Voigt notation (with sqrt(2) and 2 multipled to some terms, which simplifies the strain and stress vector notation) it is the inverse function of Un6x6(matrix) Arguments: 4rth order tensor of dimension 3x3x3x3 returns: 6x6 matrix

Functions called by this one: passage24(a)

Pymecha.**Un6x6**(*Matrice2*)

This function takes a 6x6 matrix and returns it's equivalent 4rth order symmetrical tensor it is the inverse function of SixXSix(Tensor) Arguments: 6x6 matrix Returns: 4rth order tensor of dimension 3x3x3x3

Functions called by this one: passage42(i,j)

Pymecha.**EngineerNotation**(*Vecteur*, *n*, *PrintOpt*)

this function is usefull if you want to see the mechanical properties of a material which stiffness matrix is "Vecteur" and expressed in the El_Jt_Kt_Kl_F base (see homogénéisation en mécanique des mateiraux 1 and 2 by M. Bornert T.Bretheau and P.Gilormini

arguments: Vecteur => Stiffness tensor expressed in El_Jt_Kt_Kl_F base n => axis of transverse isotropy Print-Opt => bool type variable, defines if the function will only return El of print the properties return: El (this can be changed in the code to be another characteristic and an optional argument will be added in further versions

Pymecha.**printVoigt**(*Mat*)

A simple print function that print tensor in Voigt notation (6x6 matrix)

arguments: 3x3x3x3 Tensor returns: Null

Pymecha.**print6x6**(*Mat*)

A simple print function that print tensor in modified Voigt notation (6x6 matrix)

arguments: 3x3x3x3 Tensor returns: Null

CHAPTER

FIVE

ISOTROPIC AND TRANSVERSE ISOTROPIC BASE OF VECTOR SPACE

Pymecha.**CalculJ**()
: this function calculates J vector, this vector is part of the base of the symmetrical 4rth order isotropic tensor vectorial space J is a 3x3x3x3 tensor

Pymecha.**CalculK**(*J*, *I*)
: this function calculates K vector, this vector is part of the base of the symmetrical 4rth order isotropic tensor vectorial space k is a 3x3x3x3 tensor

  it takes as arguments J (calculated by CalculJ() function and I the Identity (of the 4rth order symmetrical tensors)

Pymecha.**CalculKe**(*n*, *I*)
: this function calculates Ke vector, this vector is part of the base of the symmetrical 4rth order transverse isotropic tensor vectorial space ke is a 3x3x3x3 tensor

  it takes as arguments n, which is the Transverse isotropy axis (3x1 vector) and I the Identity (of the 4rth order symmetrical tensors)

Pymecha.**CalculJt**(*I*, *n*)
: this function calculates Jt vector, this vector is part of the base of the symmetrical 4rth order transverse isotropic tensor vectorial space Jt is a 3x3x3x3 tensor

  it takes as arguments n, which is the Transverse isotropy axis (3x1 vector) and I the Identity (of the 4rth order symmetrical tensors)

Pymecha.**CalculIt**(*n*)
: this function calculates It vector, this vector is part of the base of the symmetrical 4rth order transverse isotropic tensor vectorial space It is a 3x3x3x3 tensor

  it takes as arguments n, which is the Transverse isotropy axis (3x1 vector)

Pymecha.**CalculKt**(*Jt*, *It*)
: this function calculates Kt vector, this vector is part of the base of the symmetrical 4rth order transverse isotropic tensor vectorial space Kt is a 3x3x3x3 tensor

  it takes as arguments J (calculated by CalculJt(I,n) function and It (calculated by CalculIt(n) function

Pymecha.**CalculF**(*It*, *n*)
: this function calculates F vector, this vector is part of the base of the symmetrical 4rth order transverse isotropic tensor vectorial space F is a 3x3x3x3 tensor

  it takes as arguments n, which is the Transverse isotropy axis (3x1 vector) and It (calculated by CalculIt(n) function

Pymecha.**CalculKl**(*K*, *Kt*, *Ke*)
: this function calculates Kl vector, this vector is part of the base of the symmetrical 4rth order transverse isotropic tensor vectorial space Kl is a 3x3x3x3 tensor

it takes as arguments K (calculated by CalculK(n) function, It (calculated by CalculKt(n) function and It (calculated by CalculKe(n) function

Pymecha.**CalculEl**(*n*)

this function calculates F vector, this vector is part of the base of the symmetrical 4rth order transverse isotropic tensor vectorial space F is a 3x3x3x3 tensor

it takes as arguments n, which is the Transverse isotropy axis (3x1 vector)

Pymecha.**CalculIT**(*n*)

this function calculates IT, which is the 4rth order tensor identity restrained in the transverse plan (the normal plan to the n vector)

it takes as arguments n, which is the Transverse isotropy axis (3x1 vector) and returns a 4rth order tensor 3x3x3x3

Pymecha.**CoordExtract**(*A*, *Vecteur*)

this function takes as entry a tensor A and another named vector because it is supposed to be part of a vectorial space base. then it check if A can be written as N*Vector with N being a float.

arguments: two 4rth order tensors returns: N if A = N*Vector or 0 if not

Pymecha.**ProjectionEl_Jt_Kt_Kl_F**(*C*, *n*)

this function takes as entry a vector n, then it generates El,JT,Kt,Kl and F (see the functions named CalculEl(), CalculJT, etc.) and a stiffnes (of compliance) tensor C and finally returns a list of scalar that are the coordinates of C in the base El Jt Kt etc.

let's consider A = a*EL+b*Jt+g*F+gp*Ft+d*Kt+dp*Kl then this function will return: [A,d,dp] with A 2x2 matrix [a , gp] [g , b] this format will be named projected format

Pymecha.**unprojectEl_Jt_Kt_Kl_F**(*Vecteur*, *n*)

this function takes as argument a projected tensor (see ProjectionEl_Jt_Kt_Kl_F(C,n) for more information) and n the transverse isotropy axis and it returns the non-projected tensor (dimension 3x3x3x3)

dependencies: some function of this module are called by unprojectEl_Jt_Kt_Kl_F(): matriceordreN() all the CalculXX functions addMat()

CHAPTER

SIX

TRANSVERSE ISOTROPIC VECTOR SPACE ALGEBRA FUNCTIONS

Pymecha.**ProduitIsoTrans**(*A*, *B*)
this function takes as arguments two tensors in projected format, and returns the tensorial product of these two.

Pymecha.**SoustrIsoTrans**(*A*, *B*)
this function takes as arguments two tensors in projected format, and returns the substrat of these two (A-B).

Pymecha.**SommeIsoTrans**(*A*, *B*)
this function takes as arguments two tensors in projected format, and returns the sum of these two (A+B).

Pymecha.**InverseIsoTrans**(*A*)
this function takes as argument a tensor A in projected format, and returns the inverse of it (still in projected format)

Pymecha.**ProduitScalVectIsoTrans**(*Scal*, *Vect*)
this function takes as argument a tensor A in projected format and a scalar. It returns the the tensor multiplied by the scalar.

CHAPTER

SEVEN

STIFFNESS AND COMPLIANCE TENSOR

Pymecha.**EngValuesFromC**(*C*)

this function takes as argument a stiffness tensor C (3x3x3x3), and returns a list of it's mechanical properties (with a degree of inhomogeneity up to orthotropic)

Pymecha.**RemplissageC**(*Ex*, *Ey*, *Ez*, *nuyx*, *nuzx*, *nuzy*, *Gyz*, *Gzx*, *Gxy*)

This function takes as arguments the mechanical properties of an orthotropic material (it works for materials with lower degree of inhomogeneity) and returns the corresponding stiffness tensor (3x3x3x3)

!! This function require numpy to be installed on you computer to work properly !!

CHAPTER

EIGHT

HOMOGENIZATION METHODS

Pymecha.**Lielens**(*MTplus*, *MTmoins*, *taux*)

the Lielens homogenization model is a sort of mean between the two Mori-tanaka bounds, this functions simply takes the two bounds and makes this mean. the volume ratio is also needed as second argument.

Pymecha.**ProjectTableMatIsoTrans**(*TableMat*, *n*)

takes a list of stiffness tensors as entry and the fiber/inclusion axis n and returns a list of these tensor in projected version?

Pymecha.**VoigtBound**(*TableMat*, *Taux*)

This function do the homogenization of an heterogeneous material using the voigt homogenization model arguments: TalbeMat => list of projected stiffness tensors Taux: float representing the volume ratio between the two materials n: the fiber axis vector

returns: a list of two projected tensors which are the sup and inf bound of Mori Tanaka method.

Pymecha.**MoriTanakaLongFiber**(*TableMat*, *Taux*, *n*)

This function do the homogenization of an heterogeneous material with cylindrical inclusion arguments: TalbeMat => list of projected stiffness tensors Taux: float representing the volume ratio between the two materials n: the fiber axis vector

returns: a list of two projected tensors which are the sup and inf bound of Mori Tanaka method.

Pymecha.**MoriTanakaHillTensor**(*HillTensor*, *TableMat*, *Taux*, *n*)

This function do the homogenization of an heterogeneous material with cylindrical inclusion arguments: Hilltensor => Hill influence tensor (projected form) TalbeMat => list of projected stiffness tensors Taux: float representing the volume ratio between the two materials n: the fiber axis vector

returns: a list of two projected tensors which are the sup and inf bound of Mori Tanaka method.

Pymecha.**LielensLongFiberFromProjected**(*Mat1*, *Mat2*, *Taux*)

This function takes as entry two stiffness tensors and a volume ratio between the two materials and returns the Lielens method homogenized tensor

This function works only for cylindrical inclusions, see MoriTanakaHillTensor() to homogenize inclusion with other shape. in further version, Lielens homogenization function with custom hill tensor / Eshelby tensor will be implemented.

Pymecha.**IsoTransCfromsigmaEpsilon**(*sigma*, *epsilon*)

this function calculates the stiffness tensor from the strain and stress.

Arguments: 6x1 lists Sigma (stress) and epsilon (Strain)

returns : 6x6 matrix (Stiffness matrix in modified voigt notation)

p