Composite Plate Documentation

Release \textit{version}

John Robinson

February 18, 2016
## Contents

1 Indices and tables 3
2 How to Use Composite Plate 5
3 Modules 7

Python Module Index 15
Contents:
Indices and tables

- genindex
- modindex
- search
How to Use Composite Plate

The goal of this module is to provide you with a way to determine how a composite plate will respond to either stresses or strains. In order to do this, you must define the composite plate in the following way.

1. Define a ply material using `composite_plate.Ply`
2. Use the ply and a given orientation to define a laminae using `composite_plate.Laminae`
3. One or more of the laminae in a list to define a laminate using `composite_plate.Laminate`

Example:

```python
>>> import composite_plate.classical_plate_theory as cpt

>>> E1 = 133.44;  # Modulus in fiber direction [GPa]
>>> E2 = 8.78;    # Modulus in transverse direction [GPa]
>>> G12 = 3.254;  # Shear Modulus [GPa]
>>> nu12 = 0.26;  # Poissons Ratio (principal) [

>>> h = 6;       # Ply thickness [mm]

>>> ply = cpt.Ply(E1=E1,E2=E2,G12=G12,nu12=nu12,h=h)

>>> theta_deg = 45;  # Orientation (+ from X) [deg]

>>> theta_rad = theta_deg*np.pi/180.0;  # [rad]

>>> laminae1 = cpt.Laminae(ply, theta_rad);

>>> laminae2 = cpt.Laminae(ply,-theta_rad);

>>> laminate = composite_plate.Laminate([laminae1, laminae2])
```

After the laminate is created, you can then apply strains and see the resulting stress using `composite_plate.Laminate.applied_strain()`.

Example:

```python
>>> Epsilon = numpy.matrix([]);

>>> Kappa = numpy.matrix([]);

>>> force_and_moment_dict = laminate.applied_strain(Epsilon, Kappa)

>>> N = force_and_moment_dict('N')

>>> print("The forces are Nxx = {0}, Nyy = {1}, Nxy = {2}".format(N[1],N[2],N[3]))

>>> M = force_and_moment_dict('M')

>>> print("The moments are Mxx = {0}, Myy = {1}, Mxy = {2}".format(M[1],M[2],M[3]))
```
exception composite_plate.classical_plate_theory.CompPlateError
    Base class for exceptions in this module.

__weakref__
    list of weak references to the object (if defined)

exception composite_plate.classical_plate_theory.InputError(expr, msg)
    Exception raised for errors in the input.
    expr -- input expression in which the error occurred
    msg -- explanation of the error

class composite_plate.classical_plate_theory.IsotropicLamina(E, G, h)
    Isotropic layer to be used in a laminate
    Composites can have isotropic (uniform properties) materials in addition to composite ply layers. This class
    allows the user to define a material with equal properties in all directions and use this material to construct a
    composite plate with both types of materials. Note for isotropic materials we have the constraint $E = 2G(1+\nu)$,
    therefore only $2/3$ of $E$, $G$ and $\nu$ are required to define the material properties.

Example

>>> E = 10;  # Youngs Modulus [GPa]
>>> G = 3.3; # Shear Modulus [GPA]
>>> h = 3;   # Layer Thickness [mm]
>>> iso_layer = IsotropicLamina(E=E, G=G, h=h)

__init__(E, G, h)
    Initializes properties of the isotropic layer
    Units can be any set of self consistent units desired.

Parameters
    - E (float) – Youngs Modulus
    - G (float) – Shear Modulus
    - h (float) – Layer Thickness

S_bar
    numpy.matrix
    Compliance matrix
**Exceptions:** InputError: If any input value is not > 0

```python
def __weakref__:
    list of weak references to the object (if defined)

class composite_plate.classical_plate_theory.Laminae(ply, theta_rad):
    Composite Laminae defined for each layer in a composite.

    A laminae is an individual layer in a composite laminate. It consists of a ply rotated some angle in the composite.
    The angle is defined as the angle between the plys primary (1) axis and the composites x axis.

Example

To instantiate a laminae, you must start with a ply, and then use the ply instance and a specified angle to get a
single laminae.:

```python
code snippet
```

```python
>>> E1 = 133.44;  # Modulus in fiber direction [GPa]
>>> E2 = 8.78;  # Modulus in transverse direction [GPa]
>>> G12 = 3.254;  # Shear Modulus [GPa]
>>> nu12 = 0.26;  # Poissons Ratio (principal) [ ]
>>> h = 6;  # Ply thickness [mm]
>>> ply = Ply(E1=E1,E2=E2,G12=G12,nu12=nu12,h=h)
>>> theta_deg = 45;  # Orientation (+ from X) [deg]
>>> theta_rad = theta_deg*np.pi/180.0;  # [rad]
>>> laminae1 = Laminae(ply, theta_rad);
```

```python
def __init__(ply, theta_rad):
    Initialize the composite Laminae

    Defines an individual laminae based on the ply and orientation of the laminae x-axis to the ply 1 axis
    (theta_rad)

    Parameters
    • *ply* (*composite_plate.Ply*) – A ply object
    • *theta_rad* (*float*) – The orientation of the plys primary (1) axis to the composites x
      axis [radians]

```python
def __weakref__:
    list of weak references to the object (if defined)

class composite_plate.classical_plate_theory.Laminate(laminae_list):
    A class defining a composite Laminate. Input is as the composites laminae, the laminae thickness and stacking
    order.

    Parameters *laminae_list* (*list of composite_plate.Laminae*) – A list of laminae
    which compose the laminate, in order from bottom to top

    *A, B, D* *numpy.matrix*

    Compliance matrices will become class variables after __init__.

    *laminae_midplane_z* *list[float]*

    A list of heights from the midplane of the laminate to the midplane of each laminae.

    *Epsilon* *numpy.matrix*
If defined contains a vector [3x1] of the midplane laminate strains.

**Kappa**

*numpy.matrix*

If defined contains a vector [3x1] of the laminate curvatures.

**Example**

To instantiate a laminate, you must start with one or more laminae, and then use the laminae instance(s) in a list to create the laminate:

```python
g = 133.44; # Modulus in fiber direction [GPa]
E2 = 8.78; # Modulus in transverse direction [GPa]
G12 = 3.254; # Shear Modulus [GPa]
nu12 = 0.26; # Poissons Ratio (principal) [

> h = 6; # Ply thickness [mm]

> ply = Ply(E1=g, E2=E2, G12=G12, nu12=nu12, h=h)

> theta_deg = 45; # Orientation (+ from X) [deg]

> theta_rad = theta_deg*np.pi/180.0; # [rad]

> laminae1 = Laminae(ply, theta_rad);

> laminae2 = Laminae(ply, -theta_rad);

> laminate = Laminate([laminae1, laminae2])
```

\[
N(3\times 1) = [A](3\times 3) \cdot [\epsilon](3\times 1) + [B](3\times 3) \cdot [\kappa](3\times 1)
\]

\[
M(3\times 1) = [B](3\times 3) \cdot [\epsilon](3\times 1) + [D](3\times 3) \cdot [\kappa](3\times 1)
\]

Normal force/unit length [F/l] in x, y and xy directions:

\[
[N] = [N_{xx} N_{yy} N_{xy}]^T
\]

\[
[M] = [M_{xx} M_{yy} M_{xy}]^T
\]

-> Moment/unit length [F*l/l]

\[
[\epsilon] = [\epsilon_{xx} \epsilon_{yy} \gamma_{xy}]^T
\]

\[
[\kappa] = [\kappa_{xx} \kappa_{yy} \kappa_{xy}]^T
\]

Midplane strains indicated by \( \epsilon^0 \)

\[
[A]_{mn} = \sum_{j=1}^{N}(\bar{Q}_{mn})_j(h_j - h_{j-1})
\]

\[
[B]_{mn} = \frac{1}{2} \sum_{j=1}^{N}(\bar{Q}_{mn})_j(h_j^2 - h_{j-1}^2)
\]

\[
[D]_{mn} = \frac{1}{3} \sum_{j=1}^{N}(\bar{Q}_{mn})_j(h_j^3 - h_{j-1}^3)
\]
Laminae stiffness matrix $\bar{Q}$

$$
\begin{align*}
[c] &= [A_1] * [N] + [B_1] * [M] \\
[k] &= [C_1] * [N] + [D_1] * [M]
\end{align*}
$$

$$
\begin{align*}
[A_1] &= [A^{-1}] + [A^{-1}][B][D^{*^{-1}}][B][A^{-1}] \\
[B_1] &= -[A^{-1}][B][D^{*^{-1}}] \\
[C_1] &= -[D^{*^{-1}}][B][A^{-1}] = [B_1]^T \\
[D^{*}] &= [D] - [B][A^{-1}][B] \\
[D_1] &= [D^{*^{-1}}]
\end{align*}
$$

__init__(laminae_list)

Initialize class Laminate

Parameters

* **self** – This object instance.
* **laminae_list** – A list containing objects of type Laminae, defined in Laminae.py

Example

```python
ply1 = CompositePlate.Ply(...) laminae1 = CompositePlate.Laminae laminate = CompositePlate.Laminate([laminae1, laminae2, laminae1])
```

__weakref__

list of weak references to the object (if defined)

applied_strain(Epsilon, Kappa)

Method for applying midplane strains to the laminate instance and determine the resulting midplane stresses

Parameters

* **Epsilon** *(numpy.matrix)* – A [3x1] vector of midplane strains. $[c]_1 = \epsilon_0^{0}_{xx}$, $[c]_2 = \epsilon_0^{0}_{yy}$, $[c]_3 = \gamma_0^{0}_{xy}$
* **Kappa** *(numpy.matrix)* – A [3x1] vector of curvatures. $[k]_1 = \kappa_{xx}$, $[k]_2 = \kappa_{yy}$, $[k]_3 = \kappa_{xy}$

Returns

A dictionary containing the Normal stresses (N) and Moments (M)

N (numpy.matrix): A [3x1] vector containing the normal stresses
M (numpy.matrix): A [3x1] vector containing the resulting moments

Return type numpy.matrix

applied_stress(N, M)

Method for applying forces normalized by the section width to the laminate at the midplane.

Parameters

* **N** *(numpy.matrix)* – A [3x1] vector of normal stresses normalized by the section width $[N]_1 = N_{xx}$, $[N]_2 = N_{yy}$, $[N]_3 = N_{xy}$

**M (numpy.matrix)** – A [3x1] vector of moments normalized by the section width

\[ [M]_1 = M_{xx}, [M]_2 = M_{yy}, [M]_3 = M_{xy} \]

**Returns**

A dictionary containing the normal strains (Epsilon) and curvatures (Kappa)


**Return type** numpy.matrix

**laminae_midplane_strain (Epsilon=None, Kappa=None)**

Method for determining the laminae midplane strains in response to laminate midplane strains and curvatures.

If no laminate strain (Epsilon) or curvature (Kappa) is specified, the method will attempt to use the laminate’s current strain and curvature. If the laminate’s strain and curvature haven’t been set and no strain or curvature are specified, the method will throw an **InputError** exception.

Note: If the strain and curvature are specified for the method, it does not alter the laminate’s current state of strain and curvature. If they are not specified, the laminate’s strain and curvature state will be used, and the laminae midplane strains will be saved to the laminate state.

**Parameters**

- **Epsilon (numpy.matrix)** – A [3x1] vector of laminate midplane strains. If a value is specified, this will be assigned to the laminate as the current state of strain.

- **Kappa (numpy.matrix)** – A [3x1] vector of laminate curvatures. If a value is specified, this will be assigned to the laminate as the current state of curvature.

**Returns**

A list of strain values corresponding to the midplane strain in each laminae of the laminate.


**Return type** numpy.matrix

**laminae_stress (Epsilon=None, Kappa=None)**

Determine the stresses at the midplane of each laminae in the X & Y directions of the laminate. Also translates these to the principal 1 & 2 direction stresses of the ply.

If no laminate strain (Epsilon) or curvature (Kappa) is specified, the method will attempt to use the laminate’s current strain and curvature. If the laminate’s strain and curvature haven’t been set and no strain or curvature are specified, the method will throw an **InputError** exception.

Note: If the strain and curvature are specified for the method, it does not alter the laminate’s current state of strain and curvature. If they are not specified, the laminate’s strain and curvature state will be used, and the laminae stresses will be saved to the laminate state.

**Parameters**

- **Epsilon (numpy.matrix)** – A [3x1] vector of laminate midplane strains. If a value is specified, this will be assigned to the laminate as the current state of strain.

- **Kappa (numpy.matrix)** – A [3x1] vector of laminate curvatures. If a value is specified, this will be assigned to the laminate as the current state of curvature.

**Returns**

A list of stress values corresponding to the midplane stress in each laminae of the laminate.

Return type  numpy.matrix

class composite_plate.classical_plate_theory.Ply(E1, E2, G12, nu12, h)
Defines a ply for a composite.

A ply is the material which goes into a composite. Typically a ply will have a different modulus in the primary (E1) direction and the traverse (E2) direction. The ply will have a shear modulus (G12) and poisson's ratio (nu12), as well as a thickness (h)

Example

To instantiate a ply, you must specify the plys parameter values:

```python
>>> E1 = 133.44;  # Modulus in fiber direction [GPa]
>>> E2 = 8.78;  # Modulus in transverse direction [GPa]
>>> G12 = 3.254;  # Shear Modulus [GPa]
>>> nu12 = 0.26;  # Poissons Ratio (principal) [
>>> h = 6;  # Ply thickness [mm]
>>> ply = Ply(E1=E1,E2=E2,G12=G12,nu12=nu12,h=h)
```

__init__(E1, E2, G12, nu12, h)
Initializes the properties of the anisotropic ply.

Anisotropic ply. Need to define 4 of [E1,E2,nu12,nu21,G12] (4 of 5 independent variables). Dependent constraint: nu21/E2 = nu12/E1

Units can be any set of self consistent units desired.

Parameters

- **E1** *(float)* – Modulus in the primary direction
- **E2** *(float)* – Modulus perpendicular to the primary direction
- **G12** *(float)* – Shear Modulus
- **nu12** *(float)* – Poissons Ratio
- **h** *(float)* – Ply thickness

Exceptions: InputError: If any input value is not > 0

__weakref__
list of weak references to the object (if defined)

degree_of_isotropy()
Returns the degree of Isotropy of the Ply (E1/E2)

E1/E2 is a metric used to compare how isotropic plys are

class composite_plate.classical_plate_theory.PlyFromFiber
Estimate ply properties from Fiber and Matrix properties

Will provide estimates of various elastic and thermal ply properties based on the Fiber and matrix properties

__init__()
Initialize the PlyFromFiber object, all member variables to NaN

This initializes all possible member variables to NaN, since different estimated properties require different variables to be defined. When using, define the required fiber and matrix values for your property of interest, and then calculate the property.
Member Variables:  

- $E_f$: Elastic modulus of the fiber  
- $v_f$: Volume fraction of the ply that is fiber  
- $\nu_f$: Poisson's ratio of the fiber  
- $G_f$: Shear modulus of the fiber  
- $E_m$: Elastic modulus of the matrix  
- $v_m$: Volume fraction of the ply that is matrix  
- $\nu_m$: Poisson's ratio of the matrix  
- $G_m$: Shear modulus of the matrix  

__weakref__

list of weak references to the object (if defined)

elastic_props()

Calculates the elastic properties of the ply from the fiber and matrix properties.

composite_plate.classical_plate_theory.rotation_matrix(_theta_rad)

Rotation Matrix
C

composite_plate.classical_plate_theory,

7
Symbols

__init__() (composite_plate.classical_plate_theory.IsotropicLamina method), 7
__init__() (composite_plate.classical_plate_theory.Laminate method), 10
__init__() (composite_plate.classical_plate_theory.Laminae method), 8
__init__() (composite_plate.classical_plate_theory.Ply method), 12
__init__() (composite_plate.classical_plate_theory.PlyFromFiber method), 13
__weakref__ (composite_plate.classical_plate_theory.CompPlateError attribute), 7
__weakref__ (composite_plate.classical_plate_theory.IsotropicLamina attribute), 8
__weakref__ (composite_plate.classical_plate_theory.Laminae attribute), 8
__weakref__ (composite_plate.classical_plate_theory.Laminate attribute), 10
__weakref__ (composite_plate.classical_plate_theory.Ply attribute), 12
__weakref__ (composite_plate.classical_plate_theory.PlyFromFiber attribute), 13
applied_strain() (composite_plate.classical_plate_theory.Laminate method), 10
applied_stress() (composite_plate.classical_plate_theory.Laminate method), 10

E
elastic_props() (composite_plate.classical_plate_theory.PlyFromFiber method), 13
Epsilon (composite_plate.classical_plate_theory.Laminate attribute), 8
Error, 7
InputError, 7

K
Kappa (composite_plate.classical_plate_theory.Laminate attribute), 9

L
Laminae (class in composite_plate.classical_plate_theory), 8
laminae_midplane_strain() (composite_plate.classical_plate_theory.Laminate method), 11
laminae_midplane_z (composite_plate.classical_plate_theory.Laminate attribute), 8
laminae_stress() (composite_plate.classical_plate_theory.Laminate attribute), 11
Laminate (class in composite_plate.classical_plate_theory), 8

A
applied_strain() (composite_plate.classical_plate_theory.Laminate method), 10
applied_stress() (composite_plate.classical_plate_theory.Laminate method), 10

C
composite_plate.classical_plate_theory (module), 7
CompPlateError, 7

D
degree_of_isotropy() (composite_plate.classical_plate_theory.Ply method), 12

P
Ply (class in composite_plate.classical_plate_theory), 12
PlyFromFiber (class in composite_plate.classical_plate_theory), 12

R
rotation_matrix() (in module composite_plate.classical_plate_theory), 13
S

$S_{\text{bar}}$ (composite_plate.classical_plate_theory.IsotropicLamina attribute), 7