Implémentation du Solveur Mécanique Spectral

Correspondance entre les Équations Théoriques et le Code Python

Documentation Technique - Projet TBC Multicouche

Janvier 2026

Résumé

Ce document présente l'implémentation informatique du modèle mécanique spectral pour l'analyse thermoélastique de plaques multicouches. Il établit la correspondance rigoureuse entre les équations théoriques du document *Résolution_ équations.pdf* et leur traduction dans le code Python `core/mechanical.py`.

Table des matières

1 Introduction **2**
- 1.1 Système de Coordonnées . . . 2
- 1.2 Ansatz de Déplacement . . . 2

2 Équations d'Équilibre Volumique **2**
- 2.1 Équation 1 : Projection sur x_1 . . . 2
- 2.2 Équation 2 : Projection sur x_2 . . . 3
- 2.3 Équation 3 : Projection sur x_3 (Arrachement) . . . 3

3 Matrice Dynamique $M(\tau)$ Complète **3**

4 Résolution de l'Équation Caractéristique **4**

5 Vecteurs Propres de Déplacement et Contrainte **4**
- 5.1 Vecteur Propre de Déplacement V_r . . . 4
- 5.2 Matrice $R(\tau)$ et Vecteur Contrainte W_r . . . 4

6 Système Global et Conditions aux Limites **4**

7 Conclusion **5**

1 Introduction

Le solveur mécanique implémenté résout le problème d'équilibre thermoélastique d'une plaque multicouche soumise à un chargement thermique. La méthode utilisée est basée sur une décomposition spectrale (séries de Fourier) couplée à une résolution modale dans l'épaisseur.

1.1 Système de Coordonnées

— x_1, x_2 : directions dans le plan de la plaque
— x_3 : direction normale (à travers l'épaisseur)
— $\delta_1 = \pi/L_w$, $\delta_2 = \pi/L_w$: nombres d'onde spectraux

1.2 Ansatz de Déplacement

Les champs de déplacement sont développés sous forme modale :

$$U_j^{(k)}(x_3) = \sum_{r=1}^{6} A_r^{(k)} V_j^{(k,r)} e^{\tau_r^{(k)} x_3} \tag{1}$$

où τ_r sont les valeurs propres (modes) et $V_j^{(r)}$ les vecteurs propres de déplacement.

2 Équations d'Équilibre Volumique

Pour chaque couche k, l'équilibre statique impose $\mathrm{div}(\boldsymbol{\sigma}) = 0$, projeté selon les trois directions.

2.1 Équation 1 : Projection sur x_1

Forme Théorique (PDF Eq. 1)

$$\left(C_{55}\tau_r^2 - [C_{11}\delta_1^2 + C_{66}\delta_2^2]\right) V_1^{(r)} - (C_{12} + C_{66})\delta_1\delta_2\, V_2^{(r)} + (C_{13} + C_{55})\delta_1\tau_r\, V_3^{(r)} = F_{\mathrm{th},1} \tag{2}$$

Implémentation Python (lignes 35-38 de `get_M_matrix`) :

```
# Ligne 1 de la matrice M(tau)
M[0, 0] = C55 * tau**2 - K11          # K11 = C11*delta1^2 + C66*delta2
    ^2
M[0, 1] = -K12                         # K12 = (C12+C66)*delta1*delta2
M[0, 2] = K13_coeff * tau              # K13_coeff = (C13+C55)*delta1
```

Terme	Formule Théorique	Code Python
M_{11}	$C_{55}\tau^2 - (C_{11}\delta_1^2 + C_{66}\delta_2^2)$	`C55*tau**2 - K11`
M_{12}	$-(C_{12} + C_{66})\delta_1\delta_2$	`-K12`
M_{13}	$+(C_{13} + C_{55})\delta_1\tau$	`+K13_coeff*tau`

TABLE 1 – Correspondance Théorie/Code pour l'Équation 1

2.2 Équation 2 : Projection sur x_2

Forme Théorique (PDF Eq. 2)

$$-(C_{12}+C_{66})\delta_1\delta_2\, V_1^{(r)} + \left(C_{44}\tau_r^2 - [C_{66}\delta_1^2 + C_{22}\delta_2^2]\right) V_2^{(r)} + (C_{23}+C_{44})\delta_2\tau_r\, V_3^{(r)} = F_{\text{th},2} \quad (3)$$

Implémentation Python (lignes 40-43) :

```
# Ligne 2 de la matrice M(tau)
M[1, 0] = -K12                          # Symetrie: M21 = M12
M[1, 1] = C44 * tau**2 - K22            # K22 = C66*delta1^2 + C22*delta2
    ^2
M[1, 2] = K23_coeff * tau               # K23_coeff = (C23+C44)*delta2
```

2.3 Équation 3 : Projection sur x_3 (Arrachement)

Forme Théorique (PDF Eq. 3)

$$\boxed{-(C_{13}+C_{55})\delta_1\tau_r}\, V_1^{(r)} \boxed{-(C_{23}+C_{44})\delta_2\tau_r}\, V_2^{(r)} + \left(C_{33}\tau_r^2 - [C_{55}\delta_1^2 + C_{44}\delta_2^2]\right) V_3^{(r)} = F_{\text{th},3} \quad (4)$$

Note importante : Les termes encadrés ont des **signes négatifs**. Cette correction a été appliquée dans le code.

Implémentation Python (lignes 45-48) :

```
# Ligne 3 de la matrice M(tau)
# CORRECTION: Signes NEGATIFS selon Eq. 3 du PDF
M[2, 0] = -K13_coeff * tau    # -(C13+C55)*delta1*tau
M[2, 1] = -K23_coeff * tau    # -(C23+C44)*delta2*tau
M[2, 2] = C33 * tau**2 - K33 # K33 = C55*delta1^2 + C44*delta2^2
```

3 Matrice Dynamique $M(\tau)$ Complète

La matrice $M(\tau)$ 3×3 s'écrit sous forme matricielle :

$$M(\tau) = \begin{pmatrix} C_{55}\tau^2 - K_{11} & -K_{12} & +K_{13}\tau \\ -K_{12} & C_{44}\tau^2 - K_{22} & +K_{23}\tau \\ -K_{13}\tau & -K_{23}\tau & C_{33}\tau^2 - K_{33} \end{pmatrix} \quad (5)$$

avec les coefficients :

$$K_{11} = C_{11}\delta_1^2 + C_{66}\delta_2^2 \quad (6)$$
$$K_{12} = (C_{12} + C_{66})\delta_1\delta_2 \quad (7)$$
$$K_{13} = (C_{13} + C_{55})\delta_1 \quad (8)$$
$$K_{22} = C_{66}\delta_1^2 + C_{22}\delta_2^2 \quad (9)$$
$$K_{23} = (C_{23} + C_{44})\delta_2 \quad (10)$$
$$K_{33} = C_{55}\delta_1^2 + C_{44}\delta_2^2 \quad (11)$$

4 Résolution de l'Équation Caractéristique

Les valeurs propres τ_r sont obtenues en résolvant :

$$\det(M(\tau)) = 0 \tag{12}$$

Le déterminant est un polynôme de degré 6 en τ (pair, donc cubique en τ^2) :

$$P(\tau^2) = c_6\tau^6 + c_4\tau^4 + c_2\tau^2 + c_0 = 0 \tag{13}$$

Implémentation (`solve_characteristic_equation`, lignes 78-186) :

```
# Coefficient c6 (analytique exact)
c6 = props['C55'] * props['C44'] * props['C33']

# Coefficients c4, c2, c0 par evaluation numerique
P_0 = get_det_at_X(0)    # P(0) = c0
P_1 = get_det_at_X(1)    # P(1) = c6 + c4 + c2 + c0
P_2 = get_det_at_X(2)    # P(2) = 8c6 + 4c4 + 2c2 + c0

# Resolution du systeme lineaire pour c4, c2
b1 = P_1 - c6 - c0
b2 = P_2 - 8*c6 - c0
c4 = (b2 - 2*b1) / 2
c2 = b1 - c4

# Racines en X = tau^2
X_roots = np.roots([1, c4/c6, c2/c6, c0/c6])

# 6 racines tau = +/- sqrt(X)
tau_roots = [+sqrt(X), -sqrt(X) for X in X_roots]
```

5 Vecteurs Propres de Déplacement et Contrainte

5.1 Vecteur Propre de Déplacement V_r

Pour chaque valeur propre τ_r, le vecteur propre $V_r \in \ker(M(\tau_r))$ est calculé par la méthode des cofacteurs.

Normalisation : $V_3 = 1$ (convention du PDF)

5.2 Matrice $R(\tau)$ et Vecteur Contrainte W_r

La matrice $R(\tau)$ relie les déplacements aux contraintes transverses :

$$W_r = R(\tau_r) \cdot V_r = \begin{pmatrix} \sigma_{13} \\ \sigma_{23} \\ \sigma_{33} \end{pmatrix} \tag{14}$$

avec :

$$R(\tau) = \begin{pmatrix} C_{55}\tau & 0 & C_{55}\delta_1 \\ 0 & C_{44}\tau & C_{44}\delta_2 \\ -C_{13}\delta_1 & -C_{23}\delta_2 & C_{33}\tau \end{pmatrix} \tag{15}$$

6 Système Global et Conditions aux Limites

Le système de 27 équations (pour 3 couches) se décompose en :

Note : L'implémentation utilise une formulation 6N (18 équations pour 3 couches) car les 9 équations d'équilibre sont automatiquement satisfaites par la construction modale.

Bloc	**Description**	**Nb Eq.**	**Lignes**
1-3	Équilibre volumique (3 couches × 3 dir.)	9	Satisfait par $\det(M) = 0$
4	Interface 1-2 : Continuité U et σ	6	1-6 de K_{glob}
5	Interface 2-3 : Continuité U et σ	6	7-12 de K_{glob}
6	Bord bas ($z = 0$) : Surface libre	3	13-15 de K_{glob}
7	Bord haut ($z = H$) : Surface libre	3	16-18 de K_{glob}

TABLE 2 – Structure du système d'équations

7 Conclusion

L'implémentation dans `core/mechanical.py` respecte rigoureusement les équations du document *Résolution_ équations.pdf*. La correction des signes de M_{31} et M_{32} (termes de couplage U_3-U_1 et U_3-U_2) assure la conformité avec l'équation 3 du PDF.

La méthode modale utilisée est mathématiquement équivalente au système explicite 27×27 mais plus efficace numériquement car elle exploite la structure propre du problème.