# ANALYSE DYNAMIQUE D'UN BÂTIMENT DE GRANDE HAUTEUR À MONTRÉAL PAR MESURES DES VIBRATIONS AMBIANTES ET MODÉLISATION NUMÉRIQUE

par

Cyrielle SEYMAUX

MÉMOIRE PAR ARTICLES PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE AVEC MÉMOIRE EN GÉNIE DE LA CONSTRUCTION
M. Sc. A.

MONTRÉAL, LE 18 JANVIER 2026

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

**PRÉSENTATION DU JURY**

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

Mme. Ivanka Iordanova, présidente du jury
Département de génie de la construction à l'École de technologie supérieure

Mme Rola Assi, directrice de mémoire
Département de génie de la construction à l'École de technologie supérieure

M Ahmad Abo El Ezz, codirecteur de mémoire
Département de génie de la construction à l'École de technologie supérieure

M Guillaume Sieprawski, examinateur externe
NCK Inc. Montréal

Mme Marie-José Nollet, examinatrice interne
Département de génie de la construction à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 09 DÉCEMBRE 2025

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

## REMERCIEMENTS

Je tiens à remercier ma directrice de recherche, la professeure Rola Assi, mon co-directeur de recherche, le professeur Ahmad Abo El Ezz pour leurs conseils, leur accompagnement et leur appui tout au long de ce projet. Leur expertise a été d'un grand soutien à chaque étape de cette recherche.

Je tiens aussi à remercier la Banque Nationale d'avoir permis l'accès aux espaces commerciaux de la tour étudiée, facilitant ainsi la prise de mesures dans des conditions optimales. M Sean Tonner, gestionnaire au développement chez Broccolini, pour avoir rendu possible les prises de mesures in situ dans la partie résidentielle de la tour. M Sadri Daou, gestionnaires de l'immeuble, et son équipe, pour leur précieuse collaboration et leur soutien logistique qui ont assuré le bon déroulement de la campagne expérimentale.

Je veux souligner l'apport de Mitacs, pour le soutien financier octroyé dans le cadre d'une bourse de recherche qui a rendu possible ce projet en partenariat avec l'industrie.

Je veux aussi saluer la contribution de M Alain Déom, président de NCK Inc., ainsi que M Guillaume Sieprawski, directeur technique de recherche et ingénieur associé chez NCK Inc., pour leur confiance, soutien constant et regard bienveillant qu'ils ont porté sur mes capacités, véritables sources de motivation tout au long de cette recherche. Je remercie également l'ensemble de l'équipe de NCK Inc., dont l'appui technique et le professionnalisme ont grandement enrichi mon expérience.

Finalement, je remercie mes amis et ma famille pour leur appui inconditionnel, leur patience et leur présence bienveillante tout au long de cette aventure.

# ANALYSE DYNAMIQUE D'UN BÂTIMENT DE GRANDE HAUTEUR À MONTRÉAL PAR MESURES DES VIBRATIONS AMBIANTES ET MODÉLISATION NUMÉRIQUE

Cyrielle SEYMAUX

## RÉSUMÉ

Ce mémoire présente une analyse dynamique, à la fois expérimentale et numérique, d'un bâtiment de grande hauteur situé au centre-ville de Montréal. Une campagne de mesures de vibrations ambiantes (MVA). L'objectif est de quantifier in situ les propriétés modales, dont l'amortissement, et de vérifier la pertinence des coefficients de réduction de rigidité du béton fissuré utilisés en modélisation pour les bâtiments de grande hauteur flexibles, les cadres normatifs CSA A23.3 et CNB 2020 étant encore peu adaptés. Ainsi, à partir d'une campagne de MVA, couvrant les 58 étages du bâtiment étudié et menée au moyen de capteurs TROMINO®, les paramètres modaux (fréquences, amortissement, déformées modales) sont extraits à l'aide des méthodes de décomposition du domaine fréquentiel améliorée (DDFA) et d'identification stochastique en sous-espace (ISS). Ces données expérimentales sont ensuite comparées aux résultats d'un modèle éléments finis (MÉF) développé sous le logiciel ETABS, puis calibré à l'aide d'une analyse de sensibilité globale et d'une procédure d'ajustement incrémental. L'étude met en évidence des écarts significatifs entre les hypothèses normatives et les valeurs mesurées in situ, ce qui justifie l'ajustement de ces paramètres dans les analyses dynamiques. Cette recherche constitue la première étude de ce type à Montréal sur une tour instrumentée de cette envergure. Elle renforce la fiabilité des modèles numériques de conception en déterminant l'amortissement et les formes modales in situ, généralement difficiles à quantifier pour ces structures, et en évaluant l'écart de rigidité entre le modèle numérique et la réponse mesurée.

**Mots-clés :** vibrations ambiantes, DDFA, ISS, amortissement, section fissurée, analyse de sensibilité, calibration MÉF, bâtiment de grande hauteur

# DYNAMIC ANALYSIS OF A TALL BUILDING IN MONTREAL USING AMBIENT VIBRATION MEASUREMENTS AND NUMERICAL MODELING

Cyrielle SEYMAUX

## ABSTRACT

This thesis presents a dynamic analysis, both experimental and numerical, of a tall building located in downtown Montreal. The objective is to quantify the in-situ modal properties, including damping, and to assess the relevance of the cracked concrete stiffness reduction factors used in numerical models of flexible high-rise buildings, as the current CSA A23.3 and NBCC 2020 provisions remain poorly suited to such structures. Based on ambient vibration testing (AVT) covering all 58 floors and conducted using TROMINO® sensors installed throughout the structure, modal parameters (frequencies, damping ratios, and mode shapes) were extracted using the Enhanced Frequency Domain Decomposition (EFDD) and Stochastic Subspace Identification (SSI) methods. These experimental data were then compared with the results of a finite element model (FEM) developed in ETABS, which was calibrated using a global sensitivity analysis and an incremental updating procedure. The study highlights significant discrepancies between code-based assumptions and the in-situ measurements, justifying the adjustment of these parameters in dynamic analyses. This research represents the first study of its kind in Montreal on an instrumented high-rise of this scale. It enhances the reliability of design-stage numerical models by quantifying in-situ damping and mode shapes, typically difficult to assess for such structures, and by evaluating the stiffness gap between the numerical model and the measured structural response.

**Keywords**: ambient vibrations, EFDD, SSI, damping, cracked section, global sensitivity analysis, FEM model updating, high-rise building

## TABLE DES MATIÈRES

Page

## LISTE DES TABLEAUX

Page

## LISTE DES FIGURES

Page

## LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

| | |
|---|---|
| **ASG**<br>**GSA** | Analyse de sensibilité globale<br>Global sensitivity analysis |
| **CNBC**<br>**NBCC** | Code national du bâtiment du Canada<br>National building code of Canada |
| **DDF**<br>**FDD** | Décomposition du domaine fréquentiel<br>Frequency domain decomposition |
| **DDFA**<br>**EFDD** | Décomposition du domaine fréquentiel améliorée<br>Enhanced frequency domain decomposition |
| **DDL**<br>**DOF** | Degré de liberté<br>Degree of freedom |
| **DVS**<br>**SVD** | Décomposition des valeurs singulières<br>Singular values decomposition |
| **ÉLS**<br>**SLS** | États limites de service<br>Service limit state |
| **ÉLU**<br>**ULS** | États limites ultimes<br>Ultimate limit state |
| **ISS**<br>**SSI** | Identification stochastique en sous-espace<br>Stochastic subspace identification |
| **ISS-COV**<br>**SSI-COV** | Identification stochastique en sous-espace basée sur la covariance<br>Covariance stochastic subspace identification |

| | |
|---|---|
| **ISS-DONNÉE** | Identification stochastique en sous-espace basée sur les données |
| **SSI-DATA** | Data stochastic subspace identification |
| **MVA** | Mesure de vibration ambiante |
| **AVT** | Ambient vibration testing |
| **NTFR** | Nombre de points de la transformée de Fourier rapide |
| **NFFT** | Number of fast Fourier transform points |
| **SPG** | Système de positionnement global |
| **GPS** | Global positioning system |

## LISTE DES SYMBOLES ET UNITÉS DE MESURE

| Symbole | Description | Unités et valeurs |
|---|---|---|
| $a$ | Accélération | $m/s^2$ ou g |
| $\rho$ | Densité | $kg/m^3$ |
| $e$ | Épaisseur | mm |
| $g$ | Gravité | 9.81 $m/s^2$ |
| $E$ | Module de Young | MPa |
| $I$ | Inertie | $mm^4$ |
| $f$ | Fréquence | Hz |
| $T$ | Période | s |
| $\zeta$ | Amortissement | - |
| $\Delta t$ | Intervalle entre deux points | s |
| $F_s$ | Fréquence d'échantillonnage | Hz |
| $t_e$ | Temps de mesure | s |
| $C_{Fm,s}$ | Fréquence d'échantillonnage minimale | Hz |
| $C_{tmin}$ | Temps d'acquisition minimal | s |
| $T_{1st}$ | Période fondamentale du premier mode | s |
| $f_{1st}$ | Fréquence fondamentale du premier mode | Hz |

# INTRODUCTION

Le *Council on Tall Buildings and Urban Habitat* (2023) définit un bâtiment de grande hauteur comme une structure de plus de 14 étages, dépassant 50 m et présentant une élévation significative. Ce travail, mené en collaboration avec la firme de génie-conseil NCK Inc. (Montréal), présente une étude locale portant sur une tour de 59 étages au cœur du centre-ville montréalais. Dans les grandes villes nord-américaines comme Montréal, où le tissu urbain est dense et où les bâtiments élancés sont fréquents, la conception des tours soulève des défis particuliers liés aux vibrations ambiantes et au vent, omniprésents dans l'environnement bâti. Ces sollicitations peuvent affecter la performance structurelle et le confort des occupants.

Les vibrations ambiantes proviennent de sources externes (vent, trafic routier/ferroviaire, chantiers voisins) et de sources internes (ventilation CVC, ascenseurs, machineries, activités des occupants). Ces excitations n'altèrent pas les fréquences propres, l'amortissement, ni les déformées modales. Elles modulent plutôt l'amplitude (accélérations, dérives), ce qui permet leur identification in situ. Lorsque l'amortissement est faible, l'amplitude peut devenir significative, avec des enjeux aux états limites de service ÉLS (confort, fissuration de service) et sur certains composants non structuraux.

Par ailleurs, les charges de vent utilisées dans les modèles numériques sont souvent issues d'essais en soufflerie basés sur un amortissement normatif de 2 % (norme canadienne CSA A23.3 2014 – Section N9.2.1.2). Or, cet amortissement est souvent surestimé pour les tours de grande hauteur, comme le souligne Moayedi et al. (2015). Le commentaire I du Code national du bâtiment du Canada CNBC (2020) précise d'ailleurs que pour les bâtiments dépassant 200m, l'amortissement réel est fréquemment inférieur à 1 %. Un amortissement trop faible peut amplifier les déplacements inter-étages et les risques de résonance, tandis qu'un amortissement trop élevé peut affecter l'efficacité de la dissipation d'énergie. Cette étude se concentre ainsi sur l'évaluation réaliste de l'amortissement des bâtiments de grande hauteur, mais aussi sur les fréquences et déformées modales.

Dans le contexte de la conception des bâtiments de grande hauteur à Montréal, la problématique est que les paramètres dynamiques des modèles d'éléments finis (MÉF), amortissement et propriétés de sections fissurées, sont souvent fixés de manière approximative par les normes (CSA A23.3 2014, CNB 2020) et peu validés localement, ce qui crée un écart entre la réponse simulée et le comportement in situ sous vent et vibrations ambiantes, avec des incidences sur le confort aux ÉLS et la performance structurelle. En cas de surestimation de la rigidité, les fréquences propres sont trop élevées et la répartition modale des efforts est biaisée, ce qui conduit à sous-estimer les dérives et les accélérations aux ÉLS et à des vérifications de confort/fissuration non conservatrices. À l'inverse, une sous-estimation de la rigidité abaisse artificiellement les fréquences et majore les dérives et les accélérations, entraînant des dimensionnements sur-conservateurs. Au-delà de ces effets, une rigidité mal évaluée décale les périodes et la torsion, fausse la répartition des efforts (moments, tranchants, collecteurs/diaphragmes) et accentue les effets P–Δ. Deux sous-problématiques en découlent. D'une part, les hypothèses normatives de la norme CSA A23.3 2014 et du CNB 2020 concernant les paramètres modaux de conception des bâtiments de grande hauteur, notamment l'amortissement, ne reflètent pas toujours la réalité in situ et peuvent s'avérer non conservatrices. D'autre part, les hypothèses de la norme CSA A23.3 2014 relatives aux coefficients de section fissurée (inertie et section brute) pour des bâtiments de grande hauteur soumis au vent influencent la réponse modale numérique et entraînent un écart entre la rigidité du bâtiment in situ et celle du modèle numérique.

**Objectifs principal et secondaires**

L'objectif principal de ce mémoire est de développer, à partir de données et de résultats locaux, une méthodologie permettant de valider, dans le contexte montréalais, les hypothèses utilisées pour les analyses dynamiques des bâtiments de grande hauteur selon le cadre normatif en vigueur (CNB 2020, CSA A23.3 2014).

Objectifs spécifiques :

1. Quantifier les propriétés modales de la tour étudiée in-situ (fréquences, formes et amortissement) et discuter la valeur de l'amortissement au regard des hypothèses normatives.
2. Quantifier l'influence des paramètres (coefficients tenant compte de la section fissurée, module de Young, inertie…) sur la réponse dynamique et modale du bâtiment, afin d'évaluer et, le cas échéant, d'ajuster les hypothèses normatives et de modélisation.

**Hypothèses**

Les hypothèses du projet sont les suivantes : les analyses sont réalisées dans le domaine élastique linéaire. Les éléments structuraux du bâtiment sont considérés fissurés, même si la mise en service du bâtiment est récente et que les travaux des étages supérieurs n'étaient pas encore terminés lors des mesures. La piscine pleine du 10$^{e}$ étage est supposée ne pas agir comme amortisseur ni comme masse déplacée, et son effet est donc négligé. Enfin, la rigidité du revêtement extérieur en verre est considérée comme n'ayant pas d'impact sur la réponse dynamique globale; toutefois, sa contribution inertielle est prise en compte par l'ajout d'une masse linéique répartie sur le pourtour des dalles.

**Organisation du mémoire**

Dans un premier temps, ce mémoire présente, dans le chapitre 1, une revue de la littérature sur les normes canadiennes, les mesures de vibrations ambiantes et la calibration modale des paramètres physiques et géométriques (in-situ vs numérique). Ensuite, dans le chapitre 2, des mesures expérimentales sont réalisées sur ce cas réel à Montréal, à partir desquelles sont extraites les propriétés modales via les méthodes de décomposition du domaine fréquentiel améliorée (DDFA) et temporelle d'identification stochastique en sous-espace (ISS), ce qui permettra de quantifier l'amortissement. Ensuite, dans le chapitre 3, une comparaison est effectuée entre le comportement modal numérique et les résultats expérimentaux, afin de calibrer le modèle d'éléments finis (MÉF) et de valider certaines hypothèses normatives,

notamment en ce qui concerne les coefficients de la section fissurée et la rigidité du modèle. Enfin, le mémoire présente des conclusions et des recommandations basées sur les résultats obtenus.

**Contributions de mémoire**

Ce projet présente un intérêt particulier pour la communauté scientifique et industrielle québécoise, canadienne et internationale. Il s'agit de la première étude à Montréal portant sur une structure de grande hauteur instrumentée, contrairement aux rares recherches antérieures menées en Colombie-Britannique ou à l'international. Ce travail est également pertinent pour des firmes locales comme NCK Inc., qui souhaitent affiner leurs hypothèses de conception (sur la rigidité fissurée et l'amortissement) à l'aide de modèles calibrés sur des données réelles. Contrairement aux valeurs proposées dans les code et norme canadiens, cette étude repose sur des mesures expérimentales locales, renforçant la précision et la fiabilité des analyses dynamiques menées au Québec.

# CHAPITRE 1

# REVUE DE LITÉRATURE

Cette revue de littérature dresse l'état des connaissances sur les analyses modales des bâtiments de grande hauteur soumis aux charges de vent et présente les principales références normatives encadrant l'évaluation de leur comportement dynamique. Elle commence par examiner les hypothèses, limites et critères de conception prescrits par le CNBC 2020 et la norme CSA A23.3 2014, en particulier l'estimation de l'amortissement, la rigidité effective des éléments fissurés et les vérifications aux états limites de service (ÉLS), tout en situant ces exigences dans le panorama des normes canadiennes et internationales en vigueur. Elle aborde ensuite l'identification in situ des paramètres modaux à partir des vibrations ambiantes, indispensable à une représentation réaliste du comportement modal. Enfin, elle traite de la mise à jour/calibration des modèles d'éléments finis (MÉF), en mettant en évidence l'influence des paramètres physiques et géométriques incertains, notamment les propriétés de section fissurée, sur la réponse modale et les performances dynamiques des tours.

## 1.1 Cadre normatif canadien en dynamique des structures de grande hauteur

### 1.1.1 Hypothèses de rigidité en flexion

La norme CSA A23.3 (2014) encadre les hypothèses de modélisation numérique utilisées aux états limites de service, en particulier pour les bâtiments de grande hauteur, caractérisés par une plus grande flexibilité et une sensibilité accrue aux effets dynamiques du vent. Afin de représenter adéquatement et numériquement la rigidité effective des éléments structuraux fissurés, la norme CSA A23.3 prescrit, dans le tableau N9.2.1.2 (voir Tableau 1.1), des coefficients de réduction de la rigidité en flexion (EI) ou de l'inertie (I) à appliquer selon le type d'élément structuraux (poutres, murs, dalles).

Tableau 1.1 Facteurs de réduction de la rigidité en flexion selon la norme CSA A23.3 pour les ÉLS et ÉLU
Tiré de la norme CSA A23.3 2014

| Type d'élément | État limite de service (ÉLS) | État limite ultime (ÉLU) |
|---|---|---|
| Poutres de couplage sans renforcement diagonal | 0.50 Ig / 0.40 Ag | 0.40 Ig / 0.25 Ag |
| Poutres de couplage avec renforcement diagonal | 0.45 Ig / 0.45 Ag | 0.35 Ig / 0.40 Ag |
| Murs de cisaillement | 0.95 Ig / 0.95 Ag | 0.75 Ig / 0.75 Ag |
| Dalles avec renforcement minimum | 0.35 Ig | 0.20 Ig |
| Dalles en post-tension | 0.60 Ig | 0.45 Ig |
| Poutres (excluant les poutres couplées) | 0.50 Ig / 0.75 Ag | 0.40 Ig / 0.50 Ag |
| Colonnes | 1.0 Ig / 1.0 Ag | 0.70 Ig / 0.70 Ag |

Le Code national du bâtiment du Canada (CNBC) édition 2020, quant à lui, fixe les exigences minimales de performance structurale, en particulier pour les vérifications aux états limites de service (ÉLS). Pour les bâtiments de grande hauteur, qui présentent une flexibilité accrue et une sensibilité marquée aux effets dynamiques du vent, le code met l'accent sur la modélisation réaliste de la rigidité des éléments en béton armé et sur la prise en compte de critères de déformation et de confort vibratoire. Le commentaire I du CNBC (2020) renforce l'exigence des coefficients de la section fissuré de la norme CSA A23.3 2014 en soulignant que les modélisations trop rigides peuvent conduire, aux ÉLS, à des vibrations excessives ou à des déformations incompatibles avec le confort ou la tenue des éléments non structuraux. Le commentaire I du CNBC énonce également des critères de flèches horizontales dues au vent, recommandant de limiter la déformée latérale à H/500 (où H est la hauteur totale du bâtiment). Cette limite vise à prévenir les dommages aux cloisons, parements et composants non

structuraux. De plus, les accélérations admissibles en tête de bâtiment doivent être évaluées selon les critères de confort vibratoire définis dans la norme ISO 10137 2007.

La ligne directrice des Engénieurs et géoscientifiques de la Colombie Britanique (EGCB), intitulée « Structural Engineering Services for Tall Concrete Building Projects », version 1.0, 2022, précise à ce sujet que, pour un vent avec une période de retour de 10 ans, les accélérations maximales acceptables varient généralement entre 1.5 % (pour un bâtiment résidentiel) et 2.5% (pour un bâtiment commercial) de l'accélération de la pesanteur (g), selon le type d'occupation. Les occupations résidentielles tolèrent des niveaux plus faibles de vibration et d'accélération que les usages commerciaux ou institutionnels (information aussi trouvable dans le CNBC 2020). Ces critères sont essentiels pour s'assurer que la réponse dynamique du bâtiment ne compromet pas le confort des occupants, particulièrement aux ÉLS.

### 1.1.2 Hypothèses d'amortissement et son impact

Le CNBC (2020) précise aussi qu'un bâtiment est considéré comme très dynamiquement sensible si :

- sa fréquence fondamentale est ≤ 0.25 Hz, ou
- sa hauteur dépasse six fois sa largeur effective minimale, et s'il est occupé.

Pour ces structures, le recours à des essais en soufflerie est recommandé par la norme CSA A23.3 2014 en considérant un amortissement de 2%. Ces essais utilisent une maquette réduite et rigide instrumentée en soufflerie, reproduisant la rugosité du site et l'environnement bâti afin de mesurer pressions et forces dues au vent de façon réaliste, puis d'en déduire les charges de conception (intrants aux modèles numériques). Surtout lorsque le bâtiment est irrégulier avec une forme particulière, il peut exister des risques d'instabilités dynamiques et complexes comme le ballottement, la formation de vortex alternés, ou encore des instabilités aérodynamiques. La norme américaine *ASCE/SEI 49 – Wind Tunnel Testing for Buildings and Other Structures (*2021) sert de référence méthodologique. L'amortissement de la structure y joue un rôle clé surtout pour les vérifications aux ÉLS.

De nombreuses mesures in situ sur des bâtiments de grande hauteur, rapportées par Smith et Willford (2008), indiquent des valeurs réelles d'amortissement souvent inférieures à 1 %, particulièrement au-delà de 200 mètres de hauteur (comme illustré par la Figure 1.1 et le commentaire I de la norme canadienne CNB 2020 (CNBC - Division 4 2020)).

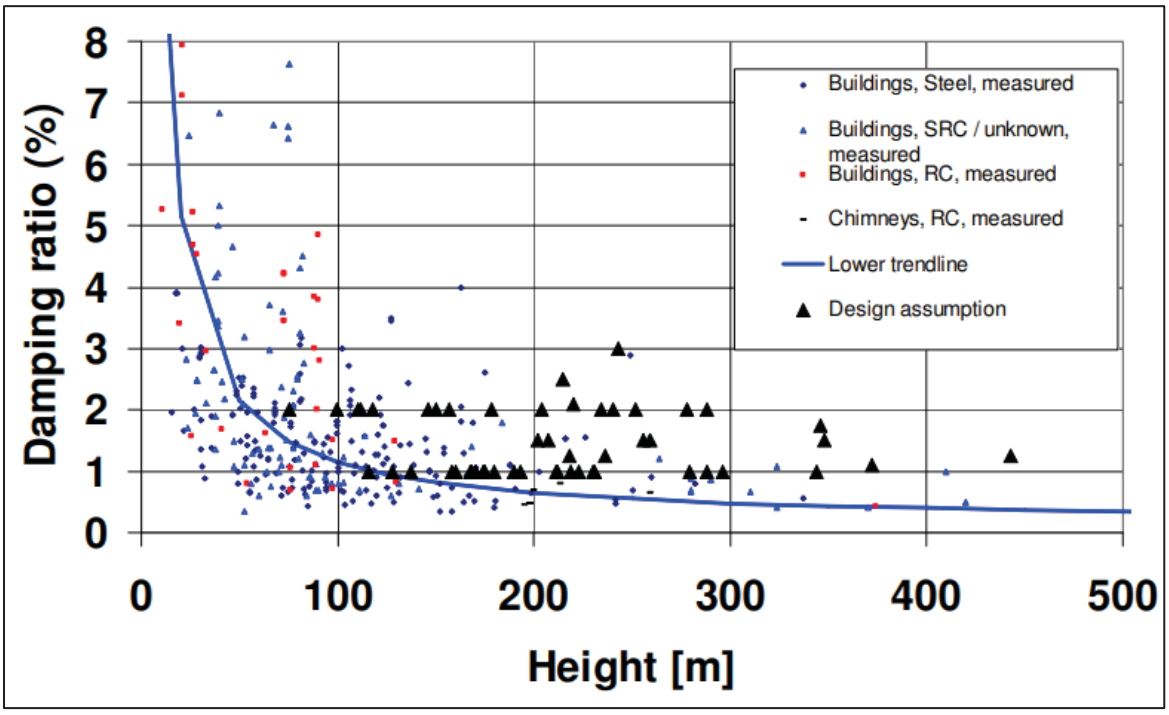



Figure 1.1 Comparaison entre les amortissements mesurés et issus des normes internationales
Tirée de Smith et Willford (2008)

Cependant, comme montré dans la Figure 1.1, les concepteurs de structures et les laboratoires d'essais en soufflerie supposent fréquemment des valeurs d'amortissement plus élevées (1 ou 2%) que celles observées en pratique. Cette approche est non conservatrice. Une sous-estimation du $\zeta$ mène à un risque de sur-conception structurelle (sections surdimensionnées, ajout d'amortisseurs). À l'inverse, quand l'amortissement supposé dépasse l'amortissement mesuré in situ, les forces de vent sont sous-estimés.

Aux ÉLUs, d'après une étude faite sur un bâtiment de 400 m de haut, il est observé que plus la valeur de l'amortissement est élevée plus le moment de renversement à la base du bâtiment

diminue (Figure 1.2, où la période t du bâtiment équivaut à 8.75 secondes). De même, la même observation est faite sur le cisaillement à la base. Aux ÉLS, cela minore les accélérations en tête de bâtiment et les déplacements inter-étage calculées, rendant les vérifications, notamment le confort des occupants, non conservatrices (Smith et Willford 2008).

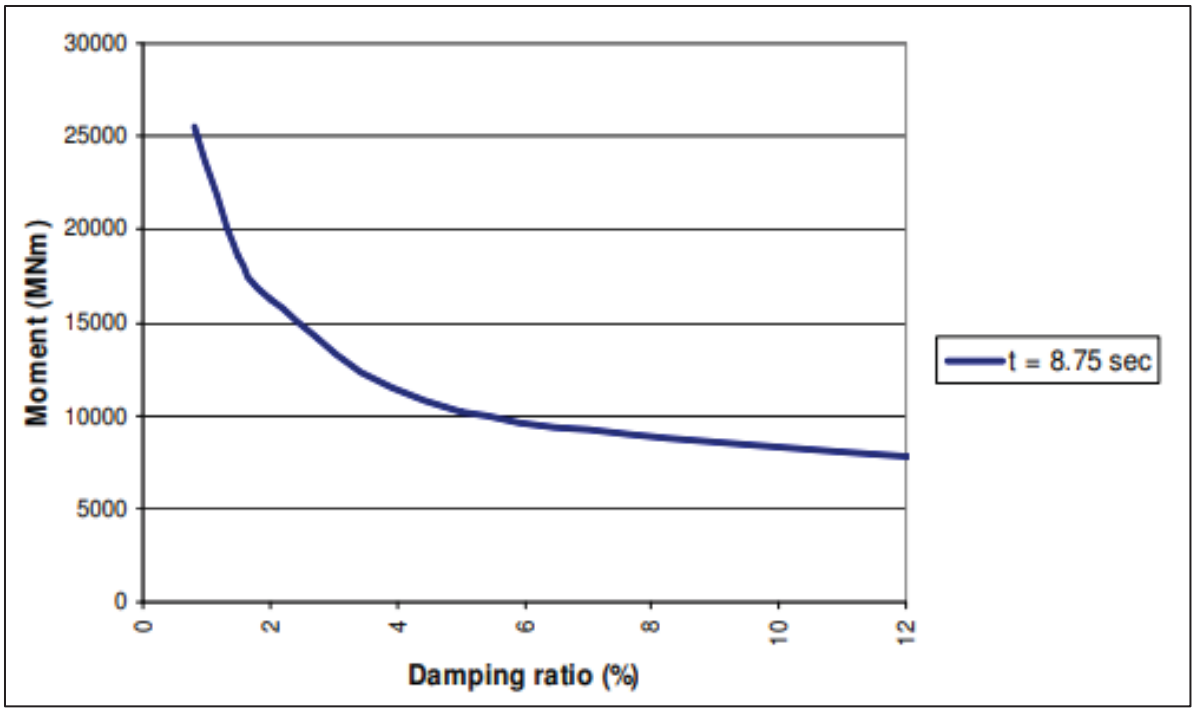



Figure 1.2 Variation du moment de renversement à la base en fonction de l'amortissement
Tirée de Smith et Willford (2008)

Plus spécifiquement, l'amortissement n'affecte pas directement les fréquences propres, il influence de manière significative l'amplitude de la réponse modale. Lorsque l'excitation (p. ex. le vent) est proche d'une fréquence naturelle, la quasi-résonance amplifie la réponse. Plus l'amortissement est faible, plus cette amplification est forte et concentrée autour de la fréquence naturelle de ses modes propres. Cette amplification se traduit par :

- des accélérations modales plus importantes,
- des vitesses et déplacements modaux accrus,
- une amplification plus marquée des efforts internes dans certains éléments structuraux.

Donc, pour les méthodes d'analyse modale fréquentielle (comme l'analyse spectrale DDFA), cela se traduit par des réponses globales plus élevées. Dans le cadre d'un modèle linéaire à amortissement proportionnel, même si les déformées modales ne changent pas avec l'amortissement, leur impact sur la réponse réelle varie en fonction de la valeur de l'amortissement introduite dans le modèle. Une même déformée modale peut induire une réponse très différente selon un amortissement de 0.8 % ou de 2 %.

De nombreux codes et normes nationaux et internationaux proposent des valeurs de référence ou des intervalles typiques pour l'amortissement à considérer dans les analyses dynamiques des bâtiments, en fonction des matériaux, des sollicitations, ou de l'objectif de l'analyse (conception aux ELU/ELS, confort ou performance). Celles-ci sont recensées dans le Tableau 1.2 ci-dessous.

Tableau 1.2 Valeurs d'amortissement utilisées selon différents codes et normes
Tiré de Singh et Mandal (2023)

| Norme / Référence | Type de bâtiment ou matériau | Amortissement critique suggéré (%) | Remarques |
|---|---|---|---|
| CSA A23.3 2014 | Béton armé (conventionnel) | 2.00 % | Valeur usuelle en conception. Pas spécifiée directement, mais couramment utilisée dans la pratique |
| CNBC 2020 - Commentaire I | Bâtiment > 200 m (mesuré in situ) | < 1.00 % | Mentionne une décroissance de l'amortissement avec la hauteur |
| ASCE 7 2022 | Bâtiment en béton (analyse dynamique au vent) | 2.00 % | 1.0 % souvent utilisé dans les études en soufflerie pour être conservateur |
| ASCE/SEI 49 2021 | Analyse en soufflerie (recommandation) | 1.00 à 1.50 % | Amortissement supérieur possible, mais nécessite une justification. |
| AS/NZ1170 2021 | Béton armé (conventionnel) | 1.00 % | - |
| EUROCODE 8 2022 | Bâtiment courant en béton | 1.57 % | Valeurs indicatives selon le matériau. Peut aller jusqu'à 5% pour les structures légères ou avec amortisseurs |
| ISO 10137 2007 | Évaluation du confort vibratoire | 1.20 %<br>Variable selon fréquence & usage | Utilisé pour fixer les seuils d'accélération en lien avec l'occupation. Ne donne pas directement de $\zeta$, mais l'implique dans les réponses attendues |
| IS 875 2016 | Béton armé (conventionnel) | 2.00 % | - |

Ces valeurs peuvent servir de point de référence initiale pour la modélisation numérique, mais doivent être utilisées avec prudence dans le cadre d'une analyse dynamique. Par exemple, une valeur usuelle de 2 % peut être utilisée pour les charges de vent en conception, mais si des mesures in situ indiquent un amortissement réel de 0.8 %, il devient nécessaire d'ajuster le modèle numérique. En effet, ceci permet de représenter fidèlement la réponse dynamique réelle de la structure, notamment pour les vérifications aux ÉLS influencées par les charges de vent. La norme d'Australie et de Nouvelle-Zélande AS/NZ 1170 (2021) génère les charges de vent les plus élevées (amortissement prescrit plus faible), tandis que la norme américaine ASCE 7 (2022) et le standard indien IS 875 Partie 3 (2016) avec une valeur d'amortissement souvent plus élevée, mènent à des charges latérales de vent minimisées (Singh et Mandal 2023). Wang et al. (2022) et Li et al. (2017) observent également une diminution de la réponse au vent latéral avec une augmentation de l'amortissement structurel.

## 1.2 Évaluation in-situ des paramètres modaux des bâtiments de grande hauteur

Cette sous-section traite de l'état des connaissances concernant l'évaluation in-situ des paramètres modaux à l'aide de mesures de vibration ambiante appliquées à des bâtiments de grande hauteur. Les méthodes d'extraction des paramètres modaux à partir de données expérimentales (méthodes fréquentielles et temporelles) sont ensuite recensées. Enfin, une synthèse de ces connaissances est faite afin que la prise de mesures pour le bâtiment de grande hauteur de ce mémoire soit adéquate.

Li et al. (2016) ont étudié deux bâtiments de grande hauteur ayant des formes irrégulières à Shanghai, Chine. Ces deux bâtiments sont présentés ci-après aux Figure 1.3 et Figure 1.4.

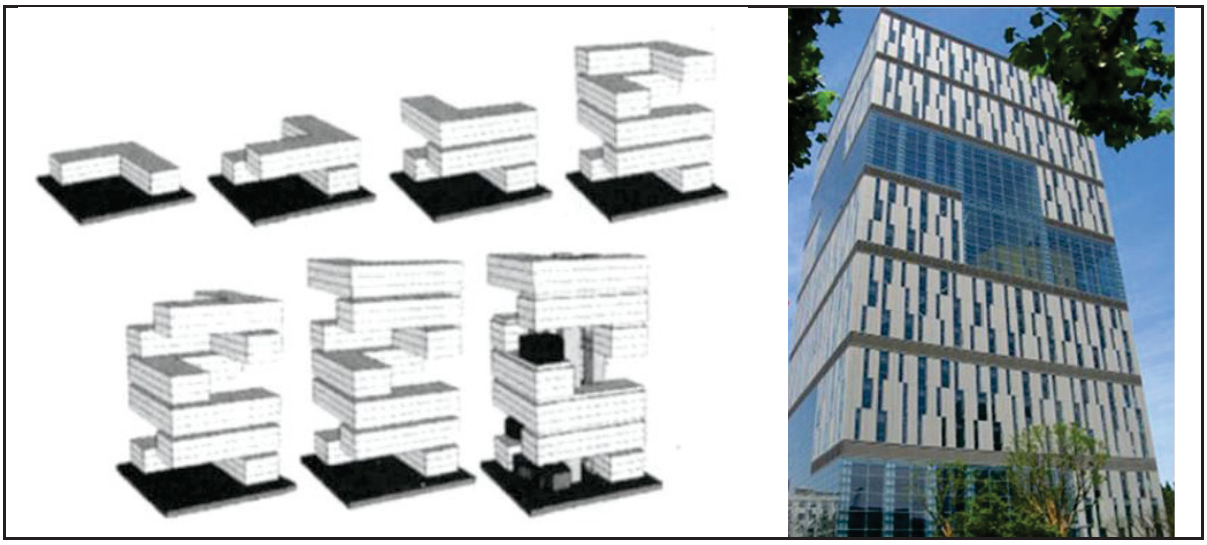


Figure 1.3 Bâtiment irrégulier en forme de L à Shanghai
Tirée de Li et al. (2016)

Le premier bâtiment (Figure 1.3) possède une hauteur totale d'approximativement 100 mètres avec une hauteur d'étage de 4 mètres. La structure est une charpente métallique composée de contreventements diagonaux en acier et d'un amortissement visqueux. Son irrégularité réside dans les étages en forme de L qui sont tournés à 90 degrés d'un étage à l'autre.


Figure 1.4 Bâtiment irrégulier en forme de H à Shanghai
Tirée de Li et al. (2016)

Le deuxième bâtiment de grande hauteur, avec 25 étages (Figure 1.4), a une irrégularité dans sa forme en H et possède un effet de couplage avec le podium de 12 étages. Ce bâtiment possède un noyau en béton armé. La hauteur de ses étages typiques est de 4 mètres. L'effet de couplage entre le podium et le reste du bâtiment est assuré par un treillis métallique. Comme les bâtiments ont une masse différente et un lien de couplage rigide (treillis), la réponse dynamique est irrégulière.

Ces bâtiments ont été instrumentés, à l'aide de vélocimètres TROMINO® ayant 10 canaux (vélocimètres et accéléromètres dans les trois directions + un SPG). Li et al. (2016, p. 89) ont utilisé le canal GPS pour synchroniser les différents appareils ensembles. Comme les instruments sont disposés sur les dalles à l'intérieur du bâtiment, le signal GPS ou radio n'a pas pu être utilisé pour les synchroniser. L'horloge interne des appareils a plutôt été utilisée pour prévenir le délai de synchronisation entre les différents appareils pendant les tests, deux pré-tests sont faits. Le premier consiste à disposer les appareils à l'air libre pour synchroniser l'horloge interne des appareils avec une fréquence d'échantillonnage de 512 Hz en utilisant leur canal de GPS. Le deuxième est basé sur l'horloge interne en laissant les appareils à la même place que lors du premier pré-test, ce qui permet de limiter le délai d'enregistrement entre les différents appareils durant toute la durée de l'expérimentation. Durant ce deuxième pré-test, une série d'impacts a été réalisée pour s'assurer que chaque appareil capture clairement les pics d'accélération ou de vitesse en fonction du temps. De sorte, l'objectif est de s'assurer qu'aucun délai d'enregistrement n'est observé entre les différents appareils (Li et al., 2016, p. 89).

Les vélocimètres, qui ont enregistré les données pendant 30 minutes voire 1 heure, dépendamment des tests réalisés, ont été placés parallèlement aux murs du noyau de cisaillement pour les deux bâtiments étudiés. Seulement un voire deux vélocimètres, selon le bâtiment étudié, sont placés sur chaque toit durant tous les tests comme illustré par la Figure 1.5.

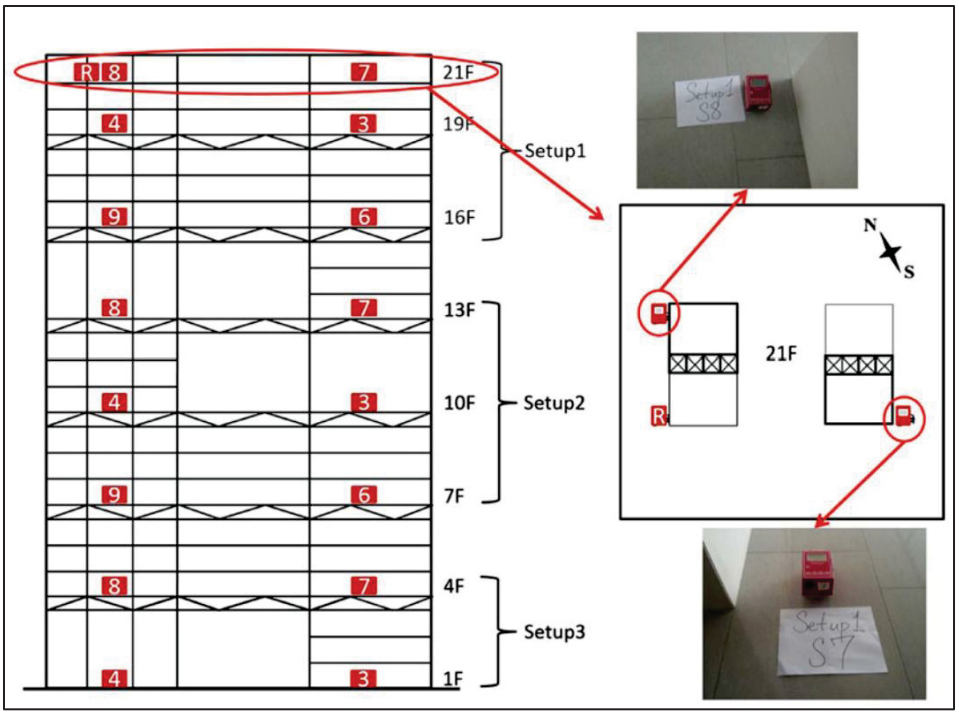



Figure 1.5 Exemple de la disposition des capteurs
Tirée de Li et al. (2016)

Les résultats obtenus démontrent que les fréquences obtenues sont similaires aux études précédentes menées sur les mêmes bâtiments. Cependant, les formes de mode n'ont pas pu être clairement identifiées, car la synchronisation de l'horodatage de chaque capteur n'a pas été précise durant les tests (même avec les pré-tests). Une autre cause est que la surface de plancher couverte par les appareils n'était pas suffisante pour capturer adéquatement la réponse en torsion des structures. Il aurait fallu disposer les appareils au niveau des extrémités de dalle. Cette étude préconise de synchroniser les vélocimètres avec le signal GPS avant chaque test pour prévenir adéquatement la dérive de l'horodatage entre les appareils.

Zhang et al. (2017) ont étudié l'une des plus grandes tours au monde : la tour de Shanghai. Cette tour dont l'allure globale est présentée par la Figure 1.6 possède 127 étages (+ 5 étages annexes) et mesure plus de 632 mètres de haut. Cette figure illustre également les éléments structuraux pour différentes zones (p. ex. en bleu : murs de cisaillement). Le revêtement extérieur est un mur rideau de forme « triangulaire » représenté par le tracé vert ci-dessous.

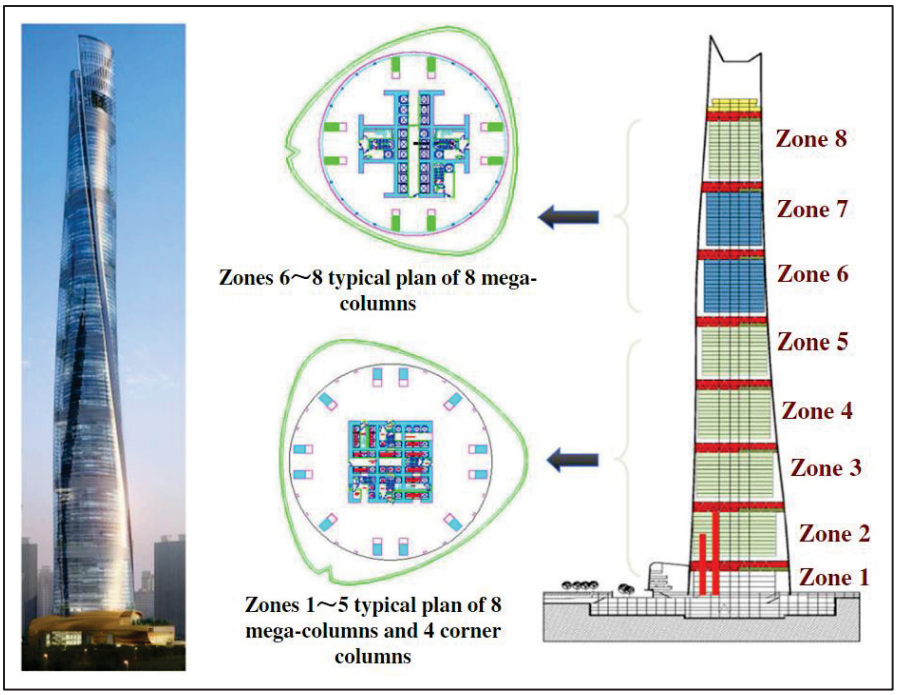
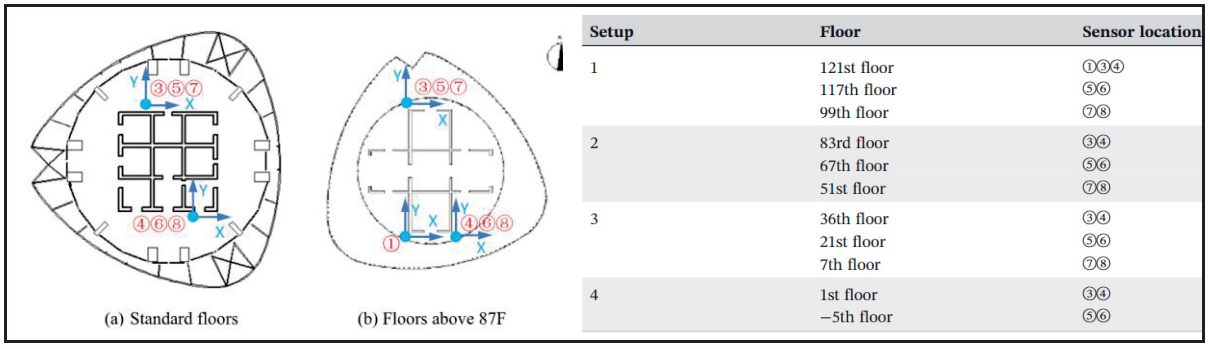



Figure 1.6 Allure globale de la tour de Shanghai
Tirée de Zhang et al. (2017)

Sa forme et hauteur intrinsèques rendent les mesures de vibrations ambiantes, pour déterminer les paramètres modaux, complexes. À noter que les vélocimètres TROMINOs® utilisés dans le cadre de cette étude seront également utilisés pour ce mémoire.




| Setup | Floor | Sensor location |
|---|---|---|
| 1 | 121st floor | ①③④ |
| | 117th floor | ⑤⑥ |
| | 99th floor | ⑦⑧ |
| 2 | 83rd floor | ③④ |
| | 67th floor | ⑤⑥ |
| | 51st floor | ⑦⑧ |
| 3 | 36th floor | ③④ |
| | 21st floor | ⑤⑥ |
| | 7th floor | ⑦⑧ |
| 4 | 1st floor | ③④ |
| | −5th floor | ⑤⑥ |

Figure 1.7 Planification des expérimentations de la tour de Shanghai
Tirée de Zhang et al. (2017)

La Figure 1.7 représente la planification et la localisation des capteurs adoptées. Il est possible de constater que les appareils sont disposés le long des murs du noyau de cisaillement. Le

senseur #1 de référence est placé en permanence au 127$^{e}$ étage (l'étage le plus en haut accessible) et restera à cette place qu'importe le test réalisé. Cette recherche met en lumière la difficulté de synchroniser les appareils de mesures, notamment entre la référence au toit et les autres étages. En effet, les signaux GPS et radio sont difficilement transmissibles à travers les murs de cisaillement et dalles en béton armée à cause du cumul d'épaisseurs difficiles à traverser pour les signaux.

La méthode proposée utilise la synchronisation manuelle par horodatage, ce qui semble être une bonne alternative pour éviter que les signaux du SPG soient bloqués par les éléments en béton armé. Pour ce faire, un premier test est fait en mettant tous les appareils les uns à côté des autres à l'extérieur du bâtiment afin de s'assurer que ceux-ci utilisent la même horloge interne grâce à la triangulation GPS. Un script de programmation numérique (MATLAB ou Python) est conçu afin d'extraire la longueur de mesure pertinente. Une fois tous les TROMINOs® en place, le délai temporel maximum au départ et minimum à la fin de chaque test est supprimé afin d'avoir des signaux ayant la même plage de temps Il est à noter qu'un enregistrement de 50 minutes est utilisé à la fréquence d'échantillonnage de 512 Hz pour limiter les biais et capturer les faibles fréquences qui se trouvent aux alentours de 0.1 Hz pour les bâtiments de très grande hauteur. Cette méthode a pu identifier les 8 modes principaux de vibrations et leurs déformées modales avec une bonne corrélation des résultats.

Turek et al. (2007, pp. 1-3) ont investigué une tour en béton armé de 44 étages dans le centre-ville de Vancouver au Canada. Cette tour résidentielle, présentée par la Figure 1.8 se compose d'un podium sur les 3 premiers étages, de 5 sous-sols, et possède un système de reprise de charges latérales composé d'un noyau.

Figure 1.8 Bâtiment régulier de grande hauteur Le Melville à Vancouver
Tirée de REW (2024)

Pour la prise de mesures, une fréquence d'échantillonnage des appareils de 500 Hz et une durée d'acquisition de 30min sont choisis. Cette fréquence est prise afin de garantir une meilleure définition des déformées modales, mais aussi pour des bâtiments de grande hauteur et irréguliers comme celui-ci. Il est d'intérêt d'acquérir le plus de données possibles quitte à les décimer par la suite.

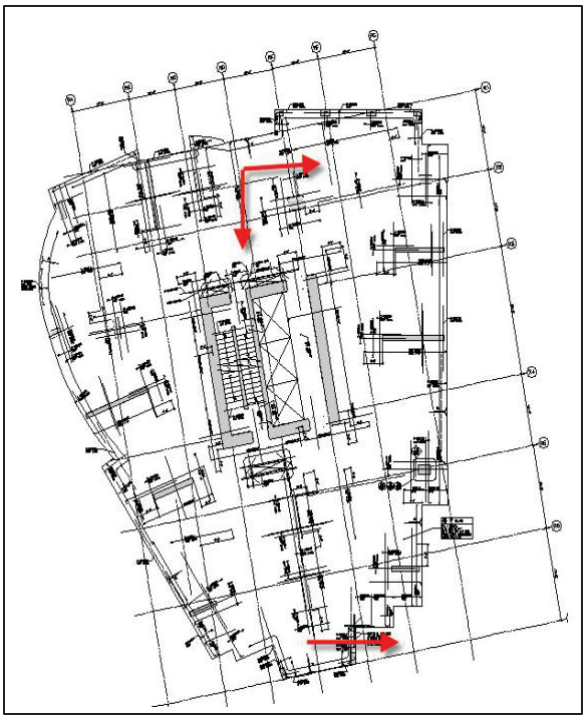


Figure 1.9 Placement des appareils sur un étage typique dans le bâtiment
Tirée de Turek et al. (2007)

Quant à la position des appareils, celle-ci est illustrée par la Figure 1.9 pour que les appareils puissent mesurer la torsion et translation aux extrémités des dalles. Il est à noter qu'un troisième appareil de référence est aussi placé à l'étage le plus haut accessible (39$^{ème}$) lors des tests. Les autres appareils connectés à la référence sont positionnés chaque 3 étages tel qu'indiqué par la Figure 1.9.

Dans cette étude, le logiciel ARTeMIS Modal Pro est utilisé avec une décimation progressive de la fréquence d'échantillonnage, d'abord à 100 Hz afin de réduire le volume de données et d'améliorer l'efficacité du traitement, puis à 5 Hz pour l'analyse modale, adaptée à la bande fréquentielle des modes d'intérêt. Les méthodes de DDFA et d'ISS sont utilisées pour extraire les paramètres modaux (fréquences, amortissements, déformées modales…).

En résumé, grâce à cette méthode, les 10 modes ont pu être identifiés correctement et sont en-dessous de 5 Hz. Les différentes méthodes utilisées dans ARTeMIs ont mené à des résultats

cohérents entre eux et des amortissements relativement constants entre 1 et 2%. Toutefois, la méthode SSI tend à fournir des résultats plus précis notamment pour l'amortissement.

## 1.3 Analyses modales pour les bâtiments de grande hauteur

Comme expliqué à la section 1.2, la méthode temporelle SSI est à privilégier pour l'évaluation des paramètres modaux, notamment de l'amortissement, dont la précision est particulièrement critique pour les bâtiments de grande hauteur. Ces structures présentent en effet des défis spécifiques liés à leur faible amortissement et à la proximité entre certaines fréquences propres, ce qui complique la séparation des modes dans les approches fréquentielles classiques. Les méthodes d'identification stochastique en sous-espace (ISS) se distinguent par leur robustesse dans ces conditions. Cette section présente les avantages des approches ISS, en comparant celles dirigées par la covariance (ISS-COV) et par les données (ISS-DONNÉE), tout en soulignant les limites de la méthode de décomposition du domaine fréquentiel améliorée (DDFA) dans ce contexte.

### 1.3.1 Les méthodes ISS-COV et ISS-DONNÉE

Les méthodes ISS se déclinent principalement en deux variantes : ISS-COV et ISS-DONNÉE. Dans les deux cas, le comportement dynamique du bâtiment est représenté par un modèle mathématique reliant les entrées inconnues et les réponses mesurées du système, mais les deux approches se distinguent par la manière dont ce modèle est estimé à partir des données expérimentales.

La méthode ISS-COV utilise les fonctions de covariance de la réponse structurelle pour construire la matrice de Hankel. Cette approche permet de filtrer naturellement le bruit en exploitant les propriétés statistiques du signal, ce qui la rend particulièrement robuste dans des conditions de faible rapport signal/bruit, comme c'est souvent le cas pour les bâtiments de grande hauteur soumis à des sollicitations environnementales faibles (Brincker et Ventura 2015). Toutefois, cette robustesse se fait au prix d'un coût computationnel plus élevé, en raison

du calcul explicite des matrices de covariance et de la taille des matrices manipulées (Reynders 2012).

À l'inverse, ISS-DONNÉE exploite directement les séries temporelles pour former les matrices d'observation. Cette méthode est plus efficace du point de vue computationnel et plus rapide à mettre en œuvre, car elle ne nécessite pas de calcul intermédiaire des fonctions de covariance (Peeters et De Roeck 2001, Reynders 2012). Cependant, elle est plus sensible au bruit, ce qui peut affecter la stabilité de l'identification modale, en particulier pour les paramètres d'amortissement (Brincker et Ventura 2015).

Plusieurs études, dont celles de Peeters et De Roeck (2001) et Magalhães et al. (2012), ont montré que, dans des conditions expérimentales idéales, les deux méthodes produisent des résultats comparables en termes de fréquences et déformées modales. Néanmoins, la méthode ISS-COV reste préférable lorsque la qualité des signaux est limitée, comme souvent pour les structures élancées ou peu excitées.

### 1.3.2 La méthode DDFA

La précision de la méthode DDFA dépend de la fréquence d'échantillonnage, du temps d'enregistrement et de la résolution. Yun et al. (2020, p. 1) ont mené une étude approfondie sur deux bâtiments de grande hauteur en Corée, dans le but d'améliorer la fiabilité de cette méthode fréquentielle. Leur travail met en évidence les conditions minimales à respecter afin d'obtenir des paramètres modaux d'une précision comparable la méthode d'ISS.

L'acronyme NTFR désigne le nombre de points utilisés pour estimer la résolution fréquentielle, c'est-à-dire le nombre d'échantillons de la transformée de Fourier rapide. Plus spécifiquement, ce nombre de points est le nombre de points utilisé pour tracer la forme du signal, soit la durée d'acquisition en seconde divisée par $\Delta t = 1$ / la fréquence d'échantillonnage $F_s$. Un NTFR insuffisant implique une mauvaise estimation de l'amortissement réel de la structure. La fréquence de résolution est donnée par :

$$fréquence\ de\ résolution = \frac{sampling\ frequency}{NTFR} \quad (1.1)$$

L'étude réalisée sur le World Financial Center de Shanghai a montré que la fraction d'amortissement obtenue à partir des vibrations ambiantes converge à mesure que la résolution fréquentielle diminue, c'est-à-dire lorsque le nombre de points NTFR augmente.

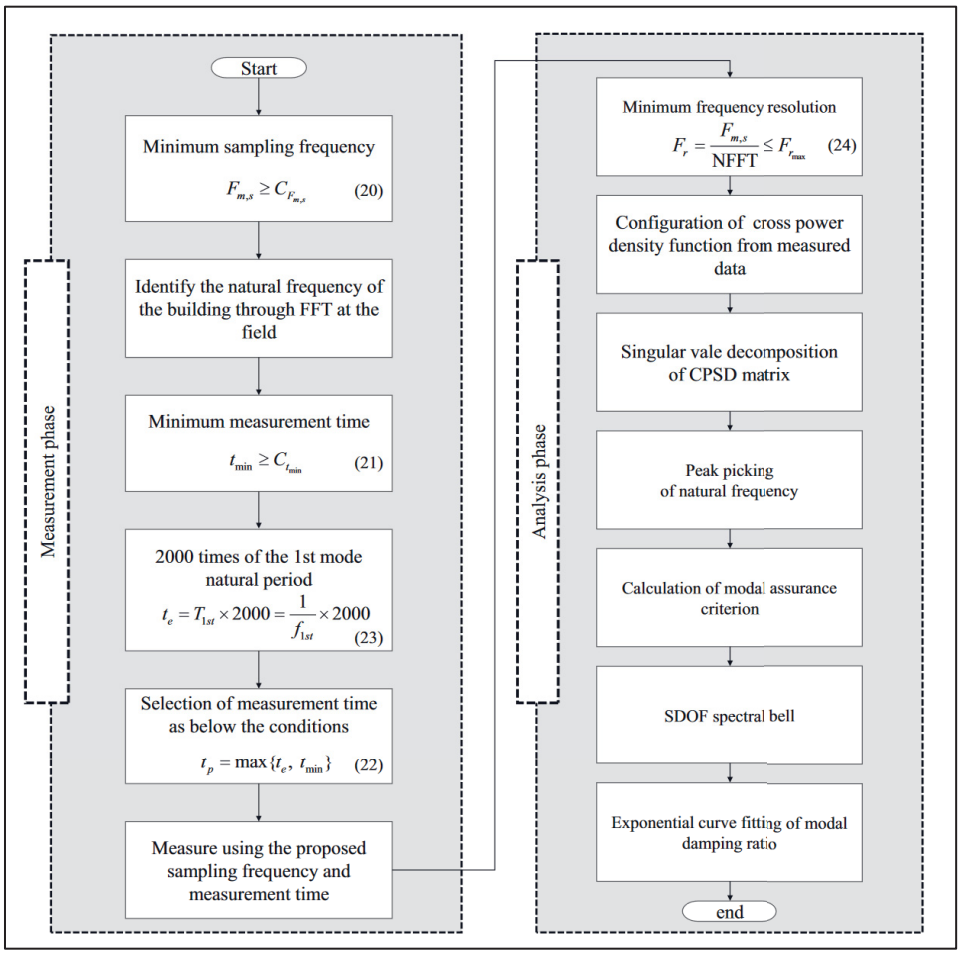



Figure 1.10 Organigramme de la méthode d'identification modale DDFA améliorée
Tirée de Yun et al. (2020)

À travers l'organigramme présenté à la Figure 1.10, les conditions minimales à considérer durant les phases de mesures expérimentales d'un bâtiment de grande hauteur sont présentées.

Ces conditions concernent le temps de mesure $t_e$, la fréquence d'échantillonnage minimum $C_{Fm,s}$ ainsi que le temps minimum $C_{tmin}$. Le temps de mesure $t_e$ est donné par :

$$t_e = T_{1st} \times 2000 = \frac{1}{f_{1st}} \times 2000 \tag{1.2}$$

Les variables $T_{1st}$ et $f_{1st}$ sont la période et fréquence fondamentale respectivement. En complément, ils recommandent d'adopter une fréquence d'échantillonnage élevée afin de faciliter l'identification des fréquences naturelles et des déformées modales. En effet, les analyses effectuées sur un modèle dynamique à quatre degrés de liberté (Figure 1.11) indiquent que la fréquence de résolution doit être inférieure à $10^{-2}$ Hz avec une fréquence d'échantillonnage d'au moins 100 Hz pour maintenir une erreur relative inférieure à 1 % lors de la détermination des modes.

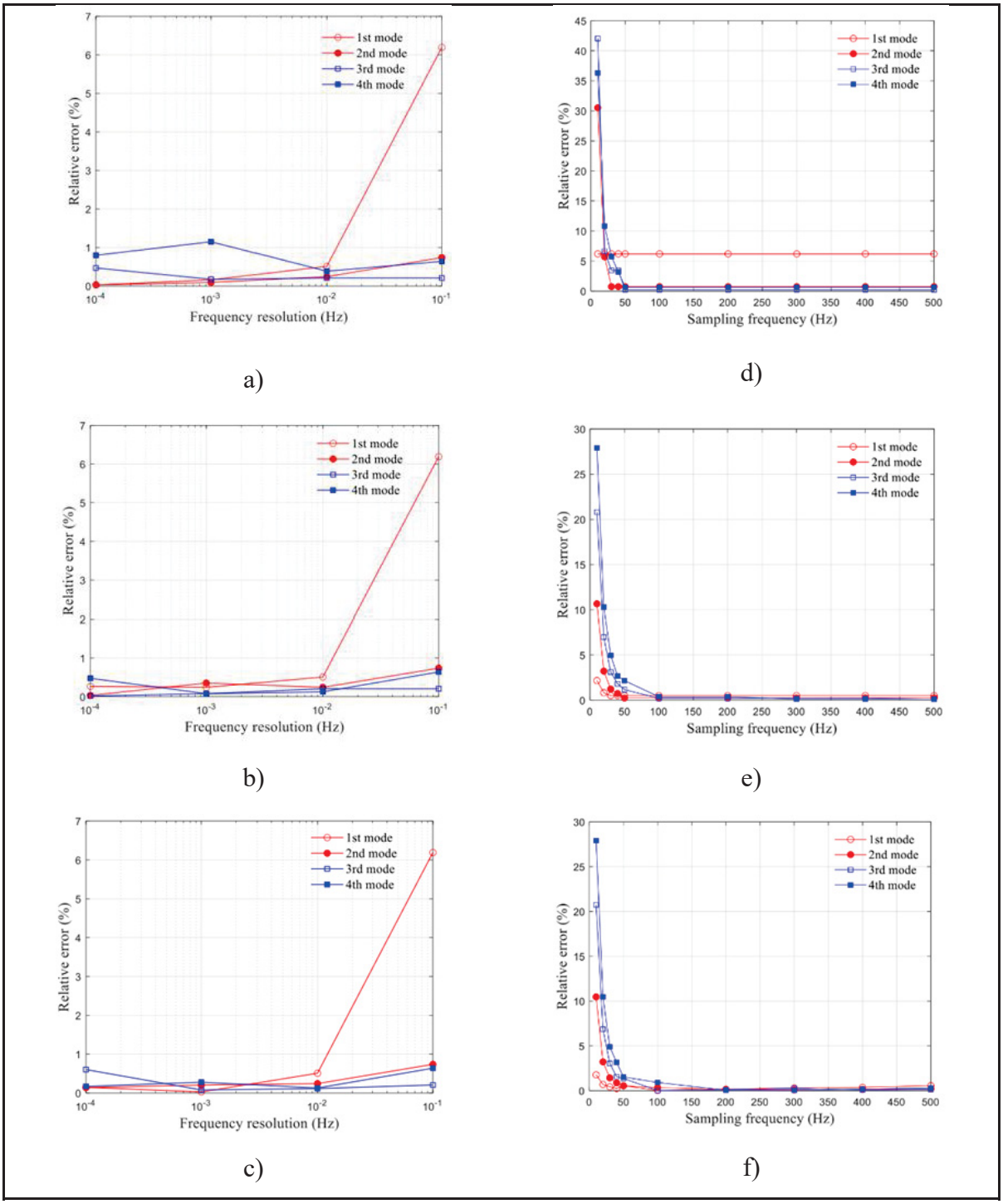



Figure 1.11 Erreur relative avec une fréquence d'échantillonnage de : a) 100, b) 200 et c) 500 Hz et des fréquences de résolution de : d) $10^{-1}$ e) $10^{-2}$ et f) $10^{-3}$ Hz
Tirée de Yun et al. (2020)

Par ailleurs, un temps d'enregistrement trop court réduit la netteté des pics spectraux, en raison d'un volume de données insuffisant, ce qui rend les modes supérieurs difficiles à identifier. De même, une fréquence d'échantillonnage trop faible peut introduire des biais significatifs (Figure 1.12).

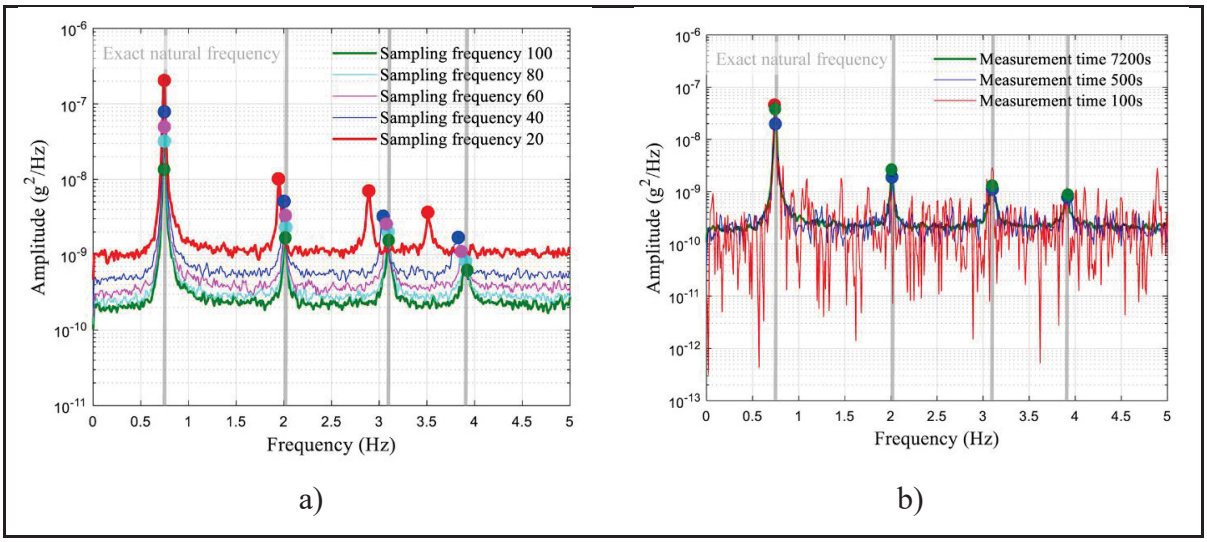



Figure 1.12 Identification des fréquences en fonction a) de fréquences d'échantillonnage et b) de temps de mesure
Tirée de Yun et al. (2020)

Pour le bâtiment de 55 étages étudié, les six premiers modes ont été correctement identifiés avec une fréquence d'échantillonnage de 500 Hz et une résolution comprise entre $10^{-2}$ Hz et $10^{-3}$ Hz. En revanche, une résolution trop fine ($2{,}5\times10^{-4}$ Hz) n'est pas recommandée, car le chevauchement des spectres entraîne des erreurs d'identification (Figure 1.13). La réduction de la fréquence d'échantillonnage, de 500 Hz à 250 Hz, divise par deux la quantité de données disponibles et accentue ce phénomène.

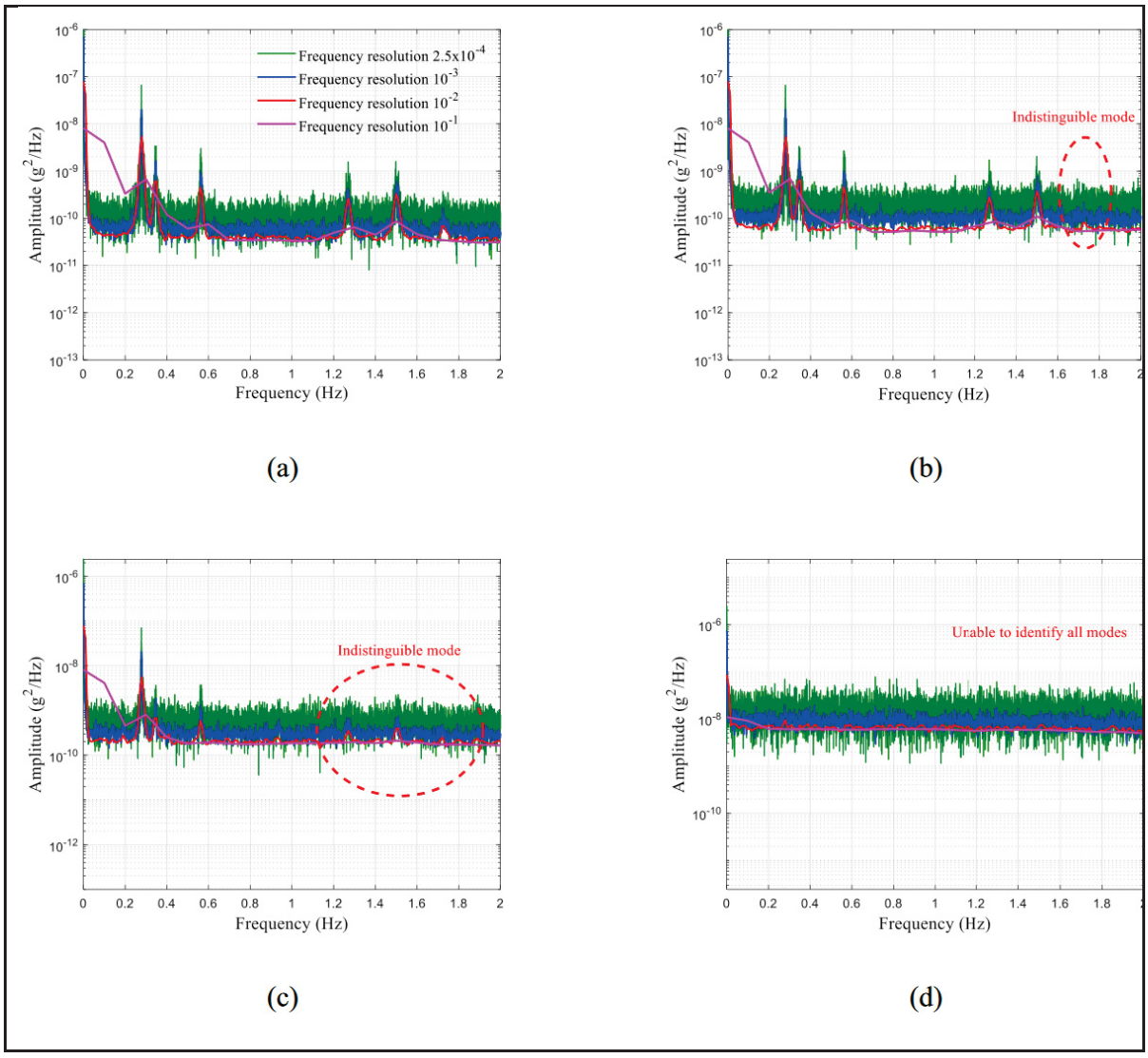



Figure 1.13 Identification des fréquences naturelles selon les fréquences d'échantillonnage : a) 500, b) 250, c) 100 et d) 50 Hz
Tirée de Yun et al. (2020)

En pratique, viser $\Delta f \in [10^{-3}, 10^{-2}]$ Hz et une fréquence d'échantillonnage $\geq 500$ Hz offre une marge confortable, et des fréquences d'échantillonnage élevées améliorent la précision des déformées modales.

Finalement, l'identification de l'amortissement utilisant la méthode DDFA est délicate. Celui-ci est estimé en ajustant la cloche (rouge) d'un système à un seul degré de liberté (DDL) à la courbe des valeurs singulières (bleu) autour du pic modal Figure 1.14. Les graphiques montrent

que l'estimation du taux d'amortissement (ζ) dépend fortement de la longueur de données (ou nombre de points de la transformée de Fourier rapide NFTR) afin d'obtenir une cloche spectrale bien dessinée.

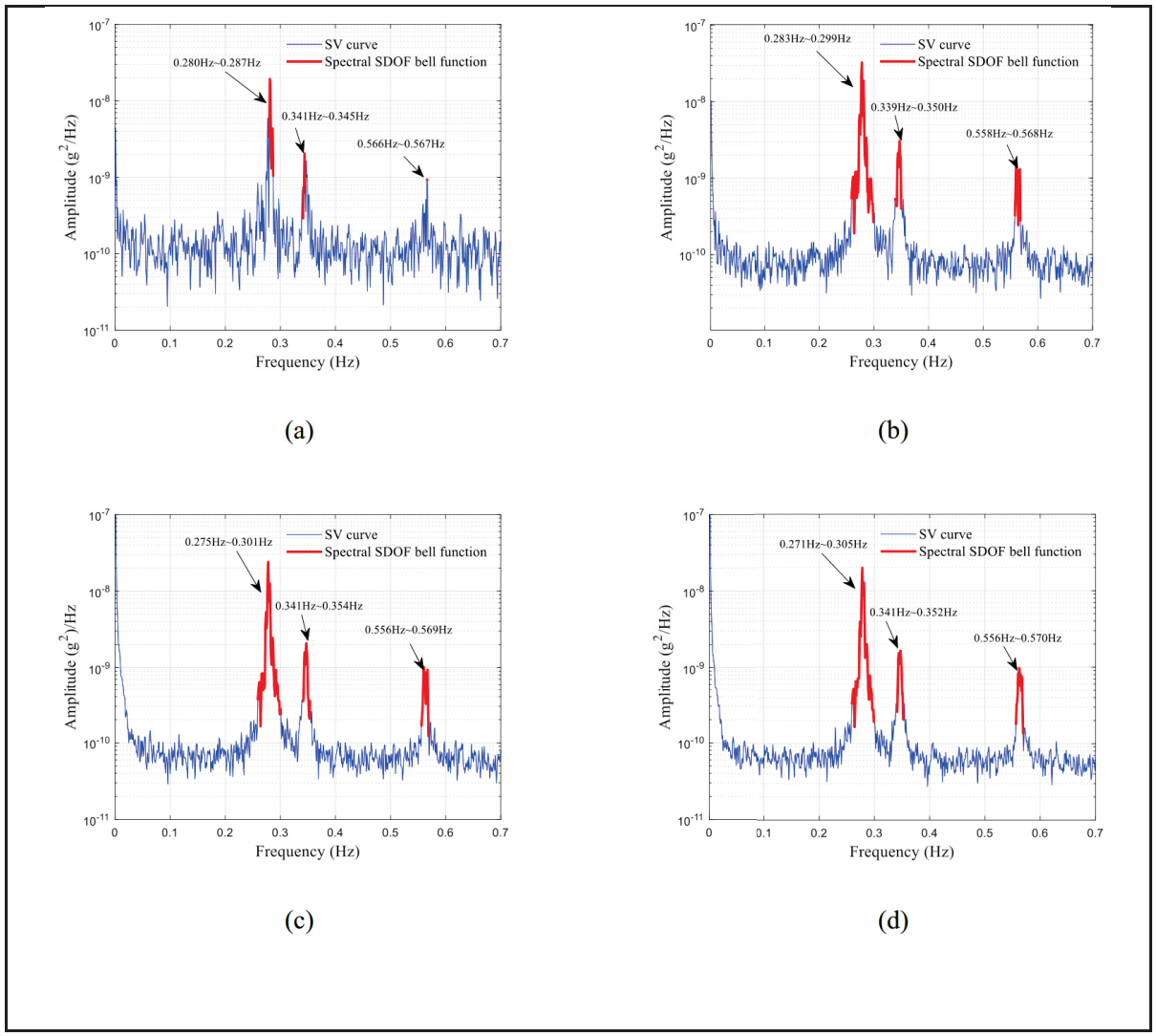



Figure 1.14 Identification de l'amortissement pour du nombre de données (NTFR) de : a) $0.5x10^6$, b) $10^6$, c) $1.5x10^6$ et d) $2.0x10^6$
Tirée de Yun et al. (2020)

Certaines règles empiriques permettent de déterminer la durée minimale d'acquisition des tests, mais elles ne garantissent pas toujours la convergence de l'amortissement. Parmi elles, il existe des seuils de $1000\times T_{1st}$ (Moayedi et al., 2015) et $2000\times T_{1st}$ (Brownjohn 2003).

Pour la tour de 55 étages, seule la règle de Brownjohn a stabilisé le taux d'amortissement ζ. Alors que pour la tour de 29 étages, aucune règle testée n'a produit un amortissement ζ stable. En pratique, il faut choisir une longueur NTFR au cas par cas et privilégier des enregistrements de plus longue durée.

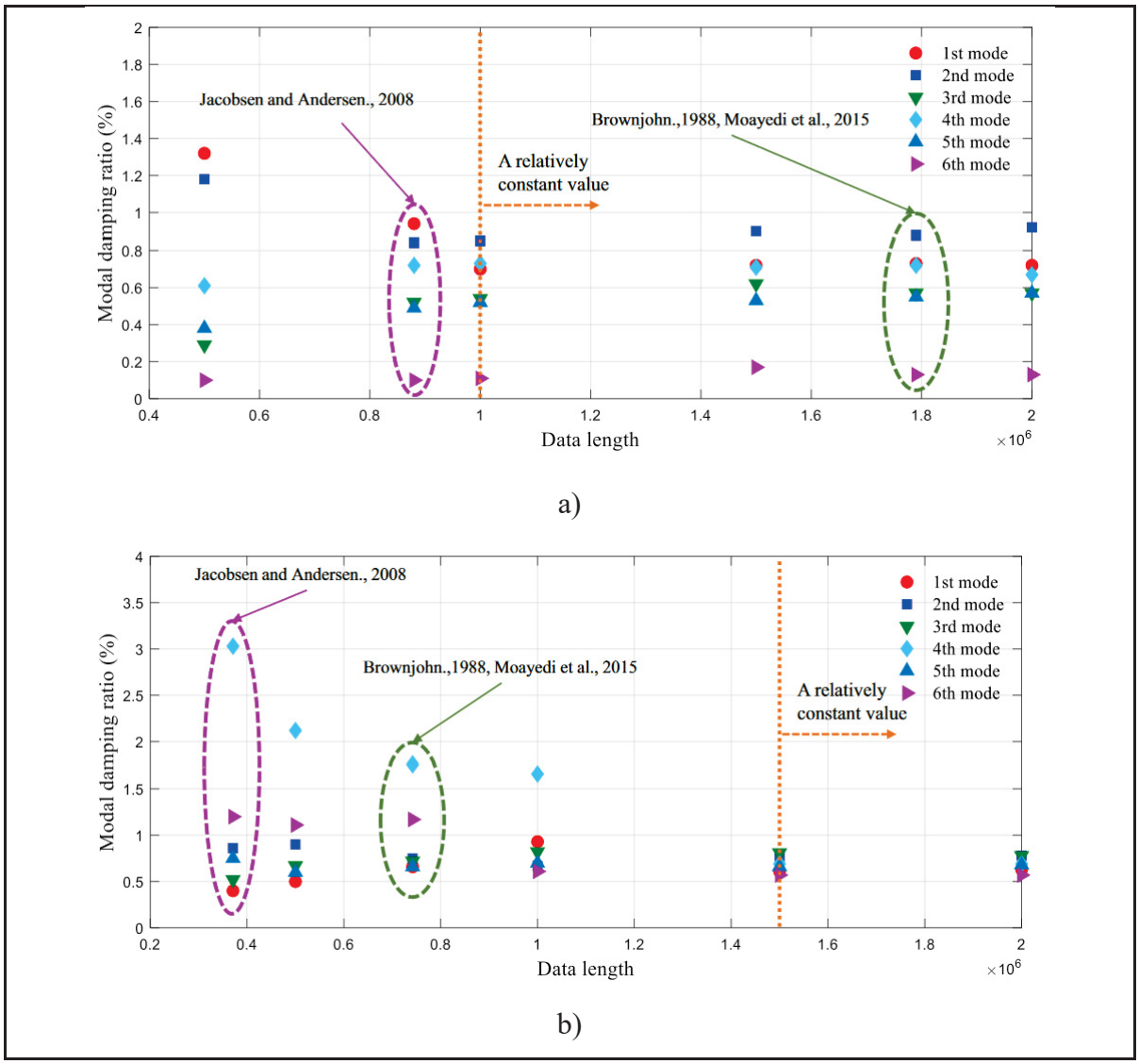



Figure 1.15 Ratio d'amortissement estimé en fonction du nombre des données (NTFR) pour un bâtiment de a) 55 étages et b) 29 étages
Tirée de Yun et al. (2020)

Ceci explique pourquoi le temps minimal d'acquisition $C_{tmin}$ est posé égal à 3000 secondes pour être conservateur et s'assurer de capturer un ratio de l'amortissement stable comme montré à la Figure 1.15. Cependant, considérer un temps d'acquisition aussi long peut être couteux en temps pour obtenir un amortissement cohérent et stable comparé aux méthodes d'ISS qui semblent être à préconiser pour les bâtiments de grande hauteur.

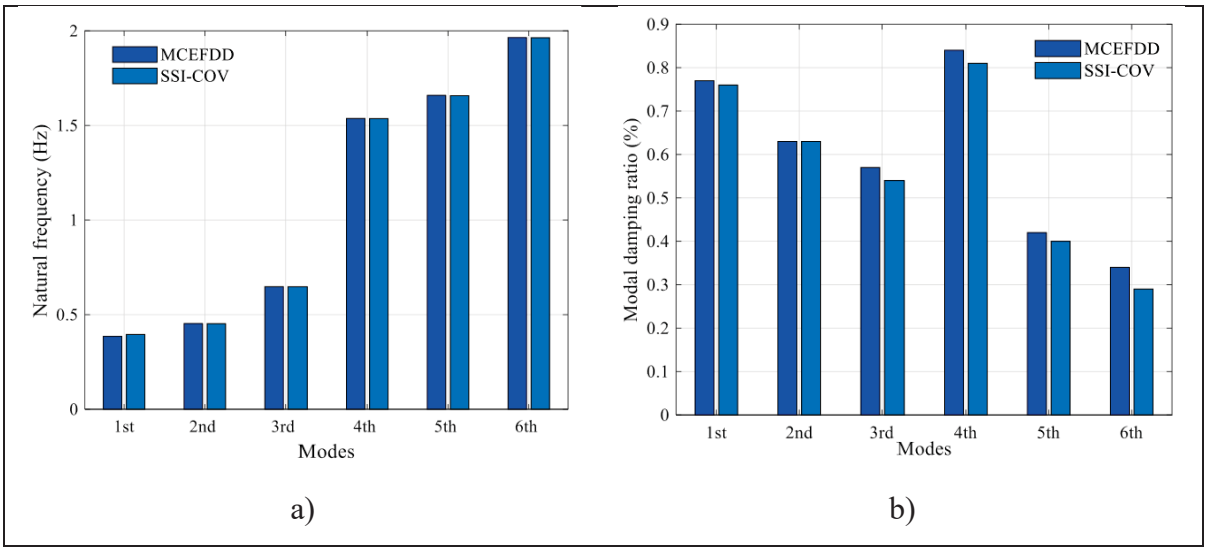



Figure 1.16 Comparaison de l'identification a) des fréquences naturelles et b) de l'amortissement utilisant la méthode décrite et d'ISS de la tour de 35 étages
Tirée de Yun et al. (2020)

Yun et al. (2020, pp. 15-16) montrent que ces conditions minimales fournissent de bons résultats proches voire similaires à ceux obtenus avec les méthodes temporelles d'identification basée sur la covariance ISS-COV (SSI-COV en anglais) ou sur les données ISS-DONNÉE (SSI-DATA) tel qu'il est illustré par la Figure 1.16.

## 1.4 Indices de qualité d'analyse modale expérimentale

Pour évaluer la fiabilité de l'identification modale in situ, des indicateurs de cohérence et de stabilité des modes ont été utilisés. Ils comparent les déformées, valident leur consistance dans un même jeu de données et quantifient la reproductibilité entre configurations. Cette section

présente trois indices courants : le critère d'assurance modale (CAM), de cohérence modale minimum (CMM) et de corrélation des déformées modales (CDM).

### 1.4.1 Critère d'assurance modale (CAM)

Le critère d'assurance modale (CAM) est le plus couramment utilisé pour comparer deux vecteurs de déformées modales. Il évalue le degré de corrélation entre deux déformées, généralement issues de différentes méthodes, configurations ou itérations d'identification. Sa valeur varie entre 0 et 1, une valeur proche de 1 indiquant une forte similarité. Bien que le CAM soit insensible à l'échelle et au signe des vecteurs, il ne détecte pas les éventuelles différences de phase ou d'orthogonalité, ce qui limite son interprétation dans certains cas complexes (Pan, et al. 2020). Le critère d'assurance modal est donné par :

$$CAM_{(\varphi_a,\varphi_e)} = \frac{|(\{\varphi_a\}^T\{\varphi_e\})|^2}{(\{\varphi_a\}^T\{\varphi_a\}\{\varphi_e\}^T\{\varphi_e\})} \tag{1.3}$$

Dans cette équation, $\varphi_a$ et $\varphi_e$ sont les déformées modales analytique et expérimentale respectivement. Tandis ce que l'indice T fait référence à la transposée du vecteur. Plus la valeur du seuil de rejet du CAM est faible, plus un grand nombre de valeurs singulières sont intégrées dans l'analyse modale. Toutefois, un seuil plus bas autorise également une plus grande déviation par rapport au vecteur de référence. Une valeur de 0,8 représente un bon compromis entre inclusion spectrale et fidélité modale (SVIBS 2025).

### 1.4.2 Cohérence modale minimum (CMM)

La cohérence modale minimum (CMM) vise à évaluer la cohérence globale d'un modèle modal. Contrairement au CAM, qui compare deux vecteurs, le CMM prend en compte l'ensemble des formes modales extraites et mesure leur compatibilité au sein du sous-espace modal estimé. Il est particulièrement utile pour détecter des modes instables ou mal identifiés et valider la structure du modèle avant son exploitation. La valeur du CMM est généralement proche de 1 lorsqu'un mode dominant est bien localisé autour d'un pic de résonance. En

revanche, elle tend à chuter de manière significative (vers 0) dans les intervalles entre les pics, où les contributions modales sont plus dispersées et mélangées. Dans ARTeMIS, l'algorithme exploite cette cohérence modale pour délimiter automatiquement les zones fréquentielles pertinentes. Le logiciel identifie ainsi le pic le plus représentatif à l'intérieur de chaque intervalle de fréquence où la cohérence modale dépasse le seuil de MMC défini par l'utilisateur (SVIBS 2025).

### 1.4.3 Corrélation des déformées modales (CDM)

La corrélation des déformées modales (CDM) évalue la cohérence d'échelle des modes en comparant leurs amplitudes relatives lorsque plusieurs estimations d'une même forme modale sont disponibles (p. ex., fenêtres temporelles ou configurations). Contrairement au CAM, insensible à l'échelle, le CDM est sensible à la normalisation et au gain et détecte ainsi des incohérences de mise à l'échelle. Intégré aux procédures d'évaluation automatique, il permet d'écarter des modes redondants ou contaminés lors d'une sélection automatique de pics en imposant un seuil CDM défini par l'utilisateur. Un CDM élevé indique une échelle stable entre les jeux de données et contribue à vérifier l'indépendance pratique entre les modes (SVIBS 2025).

### 1.4.4 Analyse modale expérimentale : synthèse et recommandations

Pour réaliser correctement des mesures de vibrations ambiantes *in situ* dans un bâtiment de grande hauteur, les TROMINO® sont positionnés de façon à capter la translation (autour du noyau de cisaillement et des cages d'ascenseurs) et la torsion (en rive de dalle). Au minimum, un appareil de référence est installé au dernier niveau accessible (idéalement en toiture) et sert de référence pour les autres capteurs afin de caractériser la translation et la torsion des déformées modales. Avant chaque essai, les appareils sont synchronisés entre eux, puis déployés à leurs points d'acquisition respectifs. L'acquisition est configurée avec une fréquence d'échantillonnage comprise entre 500 et 528 Hz et une durée d'enregistrement d'environ 3 000 x $T_{1st}$ afin d'assurer la stabilité des estimations DDFA. Cependant, pour des bâtiments très élancés, une telle durée est parfois difficile à atteindre. En pratique, de

nombreuses études (p. ex., la tour de Shanghai) retiennent environ 1 h d'acquisition. Pour l'extraction des paramètres, les méthodes temporelles SSI sont privilégiées, en particulier pour l'amortissement, tandis que la DDFA est utilisée en complément pour limiter des biais potentiels.

## 1.5 Mise à jour des modèles d'éléments finis des bâtiments de grande hauteur

La mise à jour des modèles d'éléments finis consiste à définir les paramètres physiques du bâtiment qui gouvernent la rigidité et la réponse dynamique du modèle, à mener une analyse de sensibilité pour retenir les paramètres les plus influents (Pan et al. 2020), puis à calibrer un jeu réduit de paramètres assurant un bon accord mesures versus modèle pour un coût de calcul maîtrisé (Hurtado et al., 2023). Les paramètres typiques incluent le module de Young E, l'inertie I, la densité, les épaisseurs de revêtement, des modificateurs de propriétés etc. des différents éléments structuraux (Brincker et Ventura 2015).

### 1.5.1 Analyses de sensibilité

Les méthodes locales évaluent l'effet d'un paramètre autour d'un point de référence, en ignorant les interactions. Les méthodes globales, au contraire, explorent tout l'espace de variation et captent les effets croisés. Elles sont donc privilégiées pour la sélection préalable des paramètres (Razavi et Gupta 2015, Tian 2018).

#### 1.5.1.1 Méthode de régression

Cette méthode de corrélation–régression repose sur un échantillonnage aléatoire Monte Carlo des paramètres et l'analyse des coefficients SRC, SRRC, PCC et PRCC. Les coefficients de SRC et PCC sont adaptés aux modèles linéaires des bâtiments (Tian 2018). Le coefficient standardisé SRC qui relie les paramètres physiques étudiés ($x_i$ : module d'Young, inertie, etc.) aux résultats numériques analysés (y : fréquences, déformées modales, etc.) est donné par :

$$SRC_{(y,x_i)} = b_i \frac{S_{x_i}}{S_y} \tag{1.4}$$

$S_{xi}$ et $S_y$ désignent les écarts-types des paramètres et des réponses, tandis que $b_i$ correspond au coefficient de régression associé. Le coefficient de corrélation partiel PCC est, quant à lui, obtenu par :

$$PCC_i = \frac{-C_{i_y}}{(C_{ii}C_{yy})^{1/2}} \tag{1.5}$$

Les coefficients C servant à déterminer le coefficient de corrélation s'obtiennent grâce à l'inverse de la matrice de corrélation, comme montré par :

$$C = \begin{bmatrix} r_{x_1x_1} & r_{x_1x_2} & \cdots & r_{x_1x_p} & r_{x_1y} \\ r_{x_2x_1} & r_{x_2x_2} & \cdots & r_{x_2x_p} & r_{x_2y} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ r_{x_px_1} & r_{x_px_2} & \cdots & r_{x_px_p} & r_{x_py} \\ r_{yx_1} & r_{yx_2} & \cdots & r_{yx_p} & r_{yy} \end{bmatrix}^{-1} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1p} & c_{1y} \\ c_{21} & c_{22} & \cdots & c_{2p} & c_{2y} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ c_{p1} & c_{p2} & \cdots & c_{pp} & c_{py} \\ c_{y1} & c_{y2} & \cdots & c_{yp} & c_{yy} \end{bmatrix} \tag{1.6}$$

Donc, le coefficient $r_{xi\ xj}$ est le coefficient de corrélation entre les paramètres $x_i$ et $x_j$ ; et le coefficient $r_{xi\ y}$ est le coefficient de corrélation entre les paramètres $x_i$ et les sorties du modèle y (fréquences, par exemple) (Tian 2018).

Grâce aux modules Python SciPy.stats et NumPy, il est possible de calculer les coefficients SRC et PCC. Plus la valeur absolue du coefficient SRC est élevée, plus l'influence du paramètre considéré sur la réponse du modèle est importante. Un SRC positif indique qu'une augmentation du paramètre d'entrée entraîne aussi une augmentation du paramètre modal de sortie (p. ex. la fréquence propre ou l'amplitude des déformées modales), tandis qu'un SRC négatif traduit une relation inverse. Pour le PCC, une valeur proche de ± 1 traduit une forte corrélation entre le paramètre d'entrée et le paramètre modal de sortie, contrairement à une valeur proche de 0 (Tian 2018).

#### 1.5.1.2 Méthode de criblage ou de Morris

Pour Tian (2018, 6), cette méthode consiste à fixer certains paramètres physiques (module d'Young, inertie, etc.) parmi un grand nombre, sans réduire la variance résultant de leur variation. La méthode de Morris est l'une des analyses de sensibilité globale les plus connues. Elle est qualifiée de globale, car les points de référence varient à chaque itération et les mesures de sensibilité sont obtenues à partir de la moyenne de plusieurs points évalués dans l'espace des paramètres. Ainsi, contrairement aux approches locales, elle analyse le comportement du modèle dans tout le domaine de variation des paramètres, et non seulement autour d'un point de référence.

La méthode de Morris considère les paramètres physiques d'entrée comme des valeurs discrètes (i : 1, 2, 3, 4). Elle évalue la sensibilité de chaque paramètre à partir des termes suivants :

$$\mu_i{}^* = E\left(\frac{\partial y}{\partial x_i}\right) \qquad \sigma_i{}^2 = V\left(\frac{\partial y}{\partial x_i}\right) \tag{1.7}$$

E et V désignent l'espérance et la variance respectivement. Ainsi, la moyenne μ, représente l'effet moyen du paramètre sur la fréquence de sortie, et l'écart-type σ, représente la non-linéarité ou les interactions entre les paramètres. Plus ces termes sont élevés, plus l'influence du paramètre sur la réponse du modèle (fréquences, formes modales, etc.) est importante (Razavi et Gupta 2015, 14). Il est recommandé d'utiliser la valeur absolue moyenne $\mu^*$, calculée selon :

$$\mu_i{}^* = E\left(\left|\frac{\partial y}{\partial x_i}\right|\right) \quad v_i = E\left(\left(\frac{\partial y}{\partial x_i}\right)^2\right) \tag{1.8}$$

E et V désignent toujours l'espérance et la variance respectivement. Cette méthode ne permet pas d'évaluer les interactions entre les paramètres qui influencent la réponse du modèle.

D'après Tian (2018), la méthode de Morris reste particulièrement adaptée lorsque peu de paramètres exercent une influence notable, car elle est rapide et peu coûteuse en calcul.

#### 1.5.1.3 Méthode de Sobol

Selon Razavi et Gupta (2015, p. 3082), les méthodes d'analyse basées sur la variance évaluent la variabilité de la réponse du modèle selon chaque paramètre, afin d'en déduire leur facteur de sensibilité. Contrairement à la méthode de Morris, elles permettent de quantifier non seulement l'effet individuel de chaque paramètre (dérivées partielles d'ordre 1), mais aussi leurs interactions (dérivées partielles d'ordre 2 et plus).

La variance de la réponse du système, résultant de la variation de certains paramètres, peut alors s'exprimer par :

$$V(y) = \sum_{i=1}^{n} V_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} V_{ij} + \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} V_{ijk} + \cdots + V_{1\ldots n} \tag{1.9}$$

La variance $V_i$ représente la contribution du $i^e$ paramètre à la variance totale, sans tenir compte des interactions. La variance $V_{ij}$ correspond à la contribution conjointe des paramètres $i$ et $j$, et ainsi de suite pour les autres termes. Ces variables sont définies par :

$$\begin{aligned} V_i &= V(E(y|x_i)) \\ V_{ij} &= V\left(E\left(y|x_i, x_j\right)\right) - V_i - V_j \\ V_{ijk} &= V\left(E\left(y|x_i, x_j, x_k\right)\right) - V_i - V_j - V_k - V_{ij} - V_{ik} - V_{jk} \end{aligned} \tag{1.10}$$

À noter que $E(y|x_i)$ est la valeur exceptée de y, la réponse du modèle, sachant $x_i$, un paramètre donné. Les indices de sensibilité S basés sur la variance sont définis par :

$$S_i = \frac{V_i}{V(y)}$$
$$S_{ij} = \frac{V_{ij}}{V(y)} \quad (1.11)$$
$$S_{ijk} = \frac{V_{ijk}}{V(y)}$$

Finalement, l'effet d'ordre total de l'indice de sensibilité $S_T$ est donnée par :

$$S_{Ti} = 1 - \frac{V(E(y|x_1, x_2, \dots, x_{i-1}, x_{i+1} \dots, x_n)}{V(y)} \quad (1.12)$$

$S_{Ti}$ additionne l'effet de 1$^{er}$ ordre (dérivée partielle de 1$^{er}$ ordre) du i$^{ème}$ paramètre et son interaction avec n'importe quels autres paramètres. Il est à noter que le i$^{ème}$ terme ne figure pas dans le numérateur de cet indice.

#### 1.5.1.4 Avantages et inconvénients

Selon la synthèse présentée au Tableau 1.3, chaque type d'analyse de sensibilité présente des avantages et des inconvénients selon le type de modèle et le nombre de paramètres étudiés. Le Tableau 1.3 résume les principales familles de méthodes : locales, globales ou par régression.

Tableau 1.3 Avantages et inconvénients des types d'analyse de sensibilité

Tiré de Tian (2018)

| Méthode | Sous-type | Caractéristiques |
|---|---|---|
| **Locale** | Local | Espace réduit autour des paramètres de base<br>Faible coût de calcul et simple à mettre en œuvre<br>Facile à interpréter<br>Ne considère pas les interactions entre paramètres |
| **Régression** | SRC | Modèle linéaire<br>Coût de calcul modéré<br>Rapide à calculer<br>Mesure l'influence directe de chaque paramètre<br>Ne capte pas les non-linéarités |
| | SRRC | Modèles monotones et non linéaires<br>Coût de calcul modéré<br>Rapide à calculer<br>Facile à implémenter et comprendre |
| | t-value | Modèle linéaire<br>Valeur statistique associée à la régression linéaire |
| **Globale Criblage** | Morris | Grand nombre de paramètres<br>Modèles numériques coûteux<br>Rapide à calculer |
| **Globale Variance** | Sobol | Méthode robuste basée sur la variance<br>Bon niveau de détail<br>Long à calculer |

Pour les modèles d'éléments finis complexes composés de beaucoup de paramètres, linéaires ou non linéaires, tels que ceux des bâtiments de grande hauteur, l'approche globale de type Morris offre un compromis pertinent entre précision et coût de calcul (Tian 2018).

### 1.5.2 Méthodes de calibration des bâtiments de grande hauteur

À l'issue du choix des paramètres et de l'analyse de sensibilité, le modèle d'éléments finis (MÉF) est calibré à l'aide des mesures in situ. Plusieurs méthodes de calibration permettent d'ajuster les paramètres d'un MÉF afin d'en améliorer la cohérence avec les observations expérimentales (fréquences propres et formes modales mesurées in situ). Cette section présente les principales approches pour les bâtiments de grande hauteur, des méthodes déterministes (moindres carrés, pseudo-inverse et optimisation). Les approches probabilistes comme la méthode bayésienne existent, mais ne sont pas abordées. Le choix de la méthode dépend du type de modèle, du nombre de paramètres à estimer et de la qualité des données disponibles.

#### 1.5.2.1 Exemple de la tour de Shanghai

Pan et al. (2020) ont calibré le modèle éléments finis (MÉF) de la tour de Shanghai à partir de mesures de vibrations ambiantes, afin d'évaluer l'influence des paramètres physiques et d'ajuster la rigidité du modèle numérique à celle observée in situ. La tour a été subdivisée en zones correspondant à ses différents usages (centre commercial, gymnase, résidentiel, bureaux, etc.), comme illustré à la Figure 1.17.

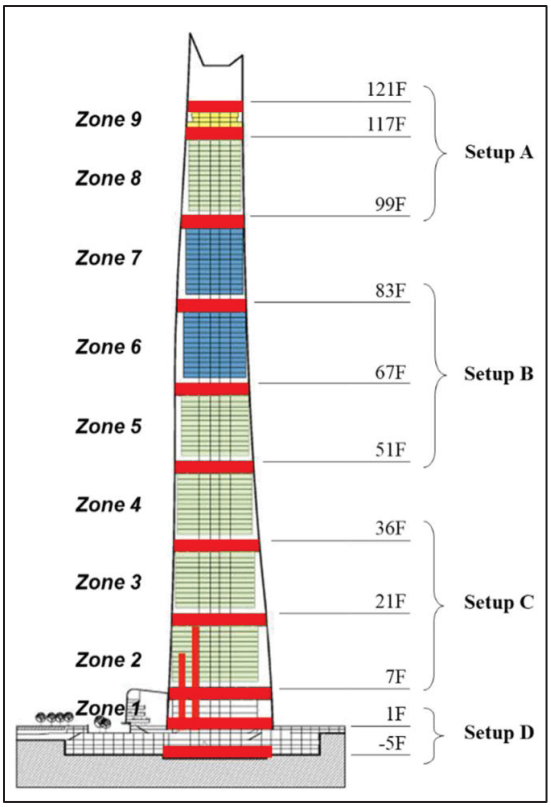



Figure 1.17 Définition des zones de la tour de Shanghai
Tirée de Pan et al. (2020)

Les paramètres choisis dans leur étude sont la densité ρ, le module d'Young E et la section A, l'inertie I en x et y, et le coefficient de torsion J des poutres de stabilisation reliant le noyau aux colonnes extérieures et stabilisatrices. L'intervalle de variation de ces paramètres est choisi comme mentionné dans le Tableau 1.4. L'analyse de sensibilité, réalisée par la méthode des différences finies, a permis d'identifier les paramètres les plus influents sur la réponse modale avant la phase de calibration. Les propriétés matérielles ont été considérées constantes au sein de chaque zone.

Tableau 1.4 Intervalle de variation des paramètres
Tiré de Pan et al. (2020)

| Paramètres | Unités | Zone structurale | Diminution | Augmentation |
|---|---|---|---|---|
| ρ | $kg/m^3$ | Zone 2-9 | -30% | +30% |
| E | MPa | Zone 1-9 | -25% | +25% |
| A | $m^2$ | Zone 2-9 | -30% | +30% |
| $I_x$ | $m^4$ | Zone 1-7 | -25% | +25% |
| $I_y$ | $m^4$ | Zone 1-7 | -25% | +25% |
| J | s.o | Zone 1-9 | -30% | +30% |

Les résultats, illustrés aux Figure 1.18 et Figure 1.19 montrent que la densité et le module d'Young des poutres exercent une influence dominante sur la réponse modale, particulièrement dans les zones 2 à 8 et pour les modes supérieurs. En revanche, les inerties $I_x$ et $I_y$ ont un effet limité. Elles influencent surtout les modes translationnels selon x et y, mais peu les modes de torsion et demeurent négligeables dans les zones supérieures (8 et 9).

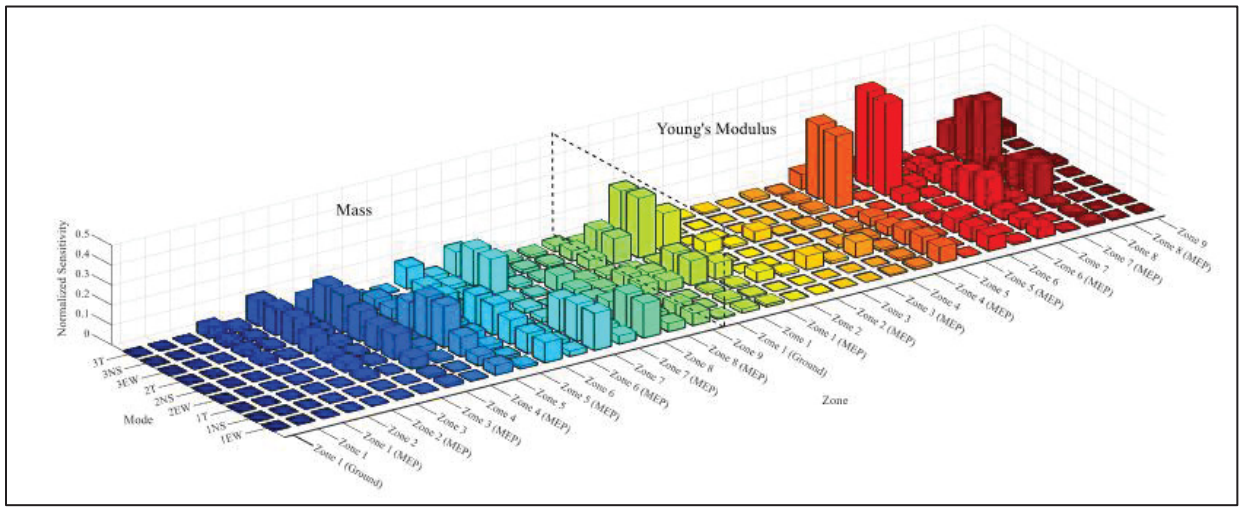



Figure 1.18 - Analyse de sensibilité de la masse et du module de Young des poutres de la tour de Shanghai
Tirée de Pan et al. (2020)

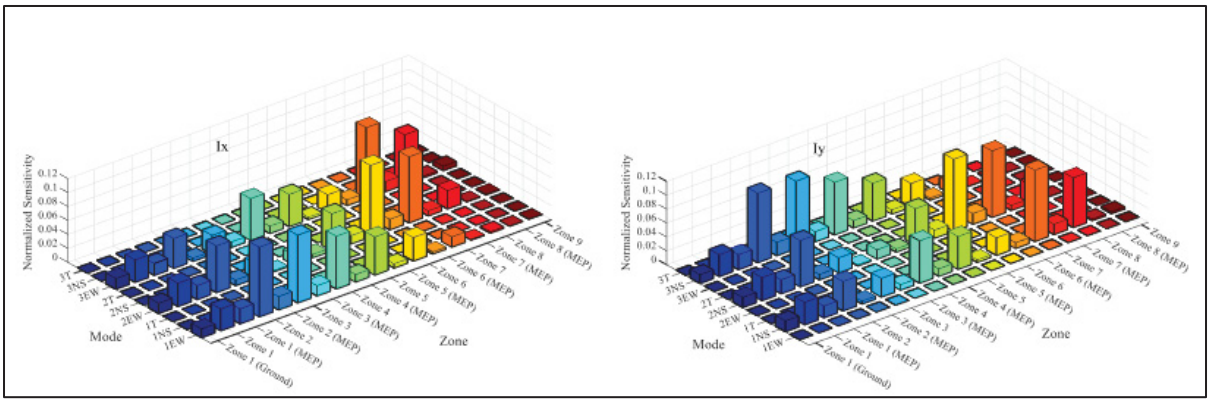



Figure 1.19 Analyse de sensibilité des inerties des poutres de la tour de Shanghai
Tirée de Pan et al. (2020)

Sur cette base, les paramètres les plus sensibles ont été retenus pour la calibration, effectuée avec le logiciel FEMTools (Dynamic Design Solutions 2025). Après environ dix itérations, la convergence a été atteinte avec une précision de 0,01 %. Les résultats indiquent une nette amélioration de la corrélation entre modèle et mesures. L'écart moyen sur les neuf premiers modes est passé de 27 % à 4 %, et les indices CAM atteignent 90 % pour les cinq premiers modes comme résumé par Tableau 1.5.

Tableau 1.5 Résultats de la calibration
Tiré de Pan et al. (2020)

| Modes | In-situ | MÉF initial | | | MÉF final | | |
|---|---|---|---|---|---|---|---|
| | Fréquence (Hz) | Fréquence (Hz) | Différence (%) | CAM (%) | Fréquence (Hz) | Différence (%) | CAM (%) |
| 1 | 0.107 | 0.113 | 5.60 | 77.7 | 0.103 | -4.40 | 97.2 |
| 2 | 0.108 | 0.120 | 11.1 | 64.7 | 0.103 | -4.60 | 98.3 |
| 3 | 0.214 | 0.311 | 45.3 | 88.1 | 0.231 | 7.90 | 92.4 |
| 4 | 0.319 | 0.389 | 21.9 | 85.1 | 0.325 | 1.90 | 90.1 |
| 5 | 0.325 | 0.390 | 20.0 | 84.9 | 0.327 | 0.60 | 90.9 |
| 6 | 0.461 | 0.644 | 39.7 | 70.7 | 0.459 | -0.40 | 84.8 |
| 7 | 0.661 | 0.881 | 33.3 | 70.4 | 0.718 | 8.60 | 76.5 |
| 8 | 0.672 | 0.887 | 32.0 | 76.0 | 0.722 | 7.40 | 79.1 |
| 9 | 0.735 | 0.983 | 33.7 | 40.1 | 0.727 | -1.10 | 75.4 |
| | | Moyenne: | 27.0 | 73.1 | Moyenne: | 4.00 | 87.2 |

La calibration et les ajustements finaux (Tableau 1.6) montrent une augmentation de la densité (+13,4 %) et de la section (+14,9 %), accompagnée d'une légère réduction du module d'Young (–8,5 %) et des inerties Ix et Iy (–17 %). Le coefficient de torsion J présente la variation la plus marquée, avec une diminution moyenne d'environ 25 %, confirmant qu'il s'agit du paramètre le plus influent lors de la calibration.

Tableau 1.6 Résultats moyen de la variation des paramètres
Tiré de Pan et al. (2020)

| Paramètres | Unités | Zone structurale | Changement moyen (%) |
|---|---|---|---|
| $\rho$ | $kg/m^3$ | Zone 2-9 | +13.4 |
| E | MPa | Zone 1-9 | -8.50 |
| A | $m^2$ | Zone 2-9 | +14.9 |
| $I_x$ | $m^4$ | Zone 1-7 | -17.4 |
| $I_y$ | $m^4$ | Zone 1-7 | -16.9 |
| J | s.o | Zone 1-9 | -25.4 |

Finalement, l'étude conclut que les bornes de variation retenues sont appropriées et que le modèle initial surestimait la rigidité de la structure, d'où des fréquences numériques supérieures aux valeurs mesurées in situ.

#### 1.5.2.2 Méthode des moindres carrés pondérés

La méthode des moindres carrés est l'une des approches les plus répandues pour la calibration des modèles éléments finis. Cette méthode repose sur la minimisation d'une fonction objective de type quadratique, exprimée en fonction du résidu entre valeurs mesurées et calculées. D'après Wu et Li (2004, p. 980), deux stratégies principales peuvent être envisagées pour cette mise à jour. Il s'agit soit de modifier directement les matrices du modèle (masse et rigidité) ou d'ajuster les paramètres physiques du modèle (matériaux et géométries). La première approche ne garantit pas une représentation précise et cohérente des propriétés physiques du bâtiment lors de la mise à jour du bâtiment. La seconde approche est donc à privilégier dans la majorité des cas, notamment pour les bâtiments de grande hauteur. Dans la formulation pondérée, deux matrices sont introduites. La matrice $W_R$, associée aux données mesurées (p. ex. fréquences, CAM) et reflétant leur variabilité/fiabilité, et la matrice $W_P$, associée aux paramètres à estimer et traduisant leur incertitude ou leur importance attendue. Cette pondération améliore la robustesse et l'interprétabilité de la mise à jour.

Le fonctionnement complet de cette méthode est illustré à la Figure 1.20 (B. Svendsen 2020)**,** qui présente l'algorithme de calibration utilisé, notamment par Wu et Li (2004). Il s'agit d'une procédure itérative. À chaque itération, une matrice de sensibilité est calculée, puis un incrément optimal $\Delta\theta$ est obtenu via un script d'optimisation Python, comme *scipy.optimize.lsq_linear*, en respectant des bornes locales et globales sur les paramètres. Une nouvelle simulation est ensuite lancée jusqu'à ce que la convergence soit atteinte.

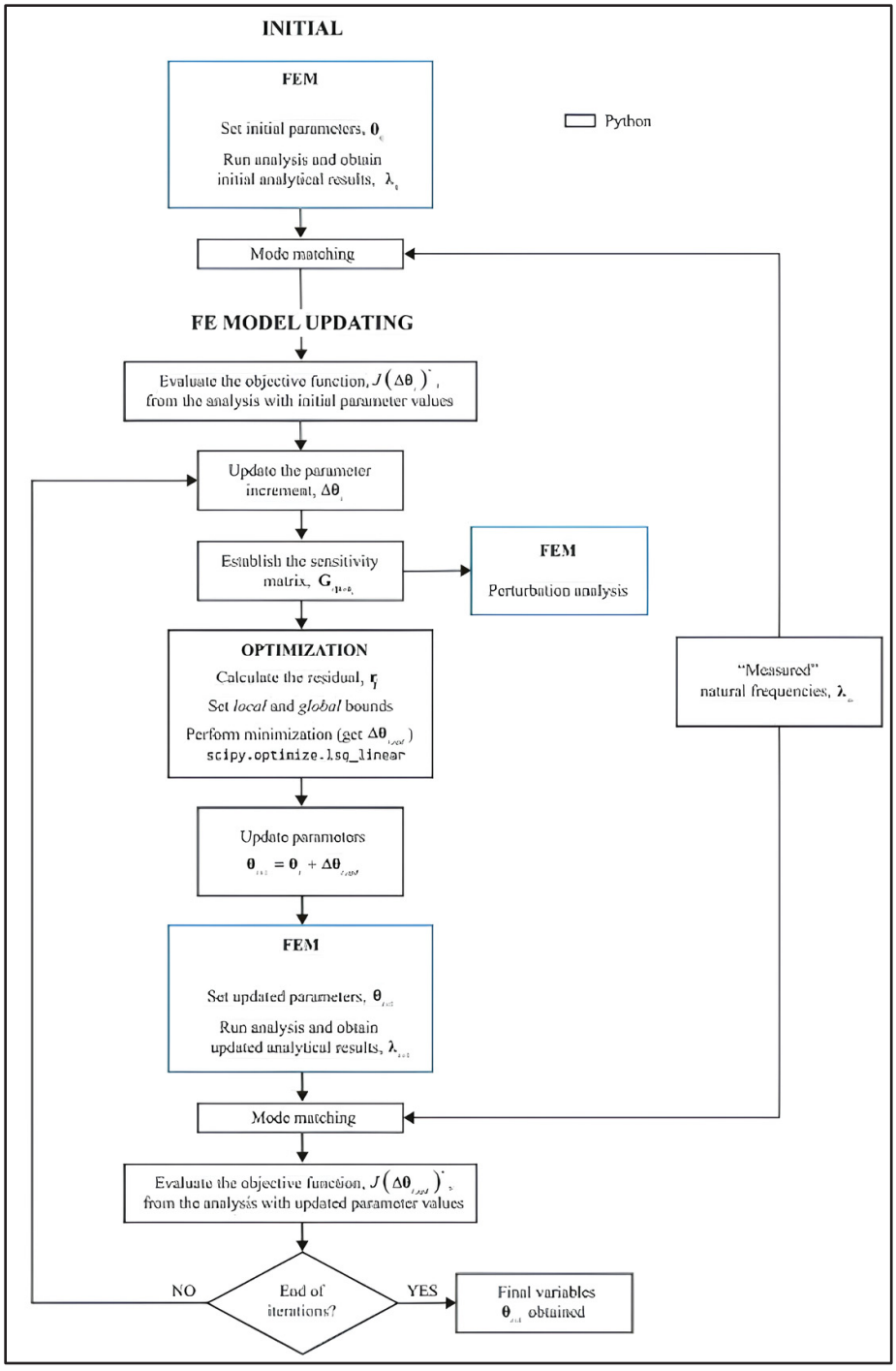



Figure 1.20 Schéma de la procédure de calibration du modèle d'éléments finis par moindres carrés pondérés
Tirée de B. Svendsen (2020)

Cette approche a été appliquée avec succès à la calibration d'un modèle d'une tour de 310 m de hauteur, dans l'étude de Wu et Li (2004), et s'est révélée plus précise que les variantes de méthodes non pondérées ou purement algébriques.

La fonction objective J(θ) est minimisée via le processus d'optimisation des paramètres, souvent définie comme l'erreur entre les résultats modaux expérimentaux et ceux du MÉF. Ces méthodes sont particulièrement utiles lorsque la relation entre les paramètres du modèle et les résultats modaux est non linéaire, lorsqu'il y a un grand nombre de paramètres ou lorsque les formulations analytiques classiques (comme la pseudo-inverse, voir ci-après) deviennent impraticables. Cependant, elles nécessitent un temps de calcul plus élevé, et leur efficacité dépend fortement de la qualité de l'initialisation, du choix des bornes, et de la construction de la fonction objectif.

#### 1.5.2.3 Méthode pseudo-inverse

La méthode par pseudo-inverse (Moore–Penrose) est une alternative courante lorsque les matrices de pondération $W_P$, $W_R$ ne sont pas définies. Elle résout la mise à jour des paramètres même pour des systèmes mal conditionnés, sous ou surdéterminés, en posant la relation linéarisée $\Delta R = S\,\Delta P$, où S est la matrice de sensibilité (réponses modales vs. paramètres) et $\Delta P$ la variation des paramètres. Cette dernière est définie comme $\Delta P = S^{+}\,\Delta R$, avec la pseudo-inverse $S^{+}$ de Moore–Penrose (par ex. $S^{+} = S^{\mathsf{T}}(SS^{\mathsf{T}})^{-1}$). Cette approche minimise $\| \Delta R - S\Delta P \|_2$, est simple à implémenter, mais sensible au bruit des données modales.

Il lui est souvent associé une régularisation ou une réduction de rang via la décomposition des valeurs singulières (DVS). Selon Wu et Li (2004, p. 985), elle est appropriée quand on veut éviter des pondérations arbitraires dans des modèles complexes. La formulation et la solution détaillées sont données par :

$$\min\ (\delta \boldsymbol{P}^{T} \delta \boldsymbol{P}) \tag{1.13}$$

$$\delta \boldsymbol{P} = \boldsymbol{S}^T(\boldsymbol{S}\boldsymbol{S}^T)^{-1}\delta \boldsymbol{R} \quad (1.14)$$

#### 1.5.2.4 Bornes d'arrêt recommandées pour l'algorithme de calibration

Pour assurer la convergence du processus de calibration et éviter l'obtention de paramètres non physiques ou numériquement instables, il est nécessaire de définir des critères d'arrêt encadrant l'évolution des paramètres mis à jour. Wu et Li (2004, p. 985) établissent des critères d'arrêt qui permettent d'éviter de large variation dans la mise à jour des paramètres qui pourrait mener à des valeurs physiques non représentatives.

L'écart entre les fréquences numériques mises à jour $f_a^{(k)}$ à l'itération k et les fréquences expérimentales $f_e$ ne doit pas dépasser une borne tolérable $| f_a^{(k)} - f_e | \leq \Delta f_{tol.}$ Dans la littérature, une tolérance de 3 à 5 % de la fréquence mesurée est couramment acceptée pour garantir une bonne convergence (Teughels, Maeck et De Roeck 2002, Mottershead, Link et Friswell 2011).

Chaque paramètre $P^{(k)}$ mis à jour à l'itération k doit rester dans une plage de valeurs physiquement réalistes, délimitée par une borne inférieure $P_{min}$ et une borne supérieure $P_{max}$. Ces bornes sont définies à partir des normes de conception (ex. : CSA A23.3 2014 pour le béton), de plages typiques issues de la littérature scientifique, ou encore à partir de valeurs extraites du modèle architectural initial (Tableau 1.7).

Tableau 1.7 Exemples de bornes utilisées pour la calibration selon la littérature

| Paramètre | $P_{mim}$ | $P_{max}$ | Référence technique |
|---|---|---|---|
| Module de Young du béton | 20 000 MPa | 40 000 MPa | (CSA Group - A23.3-14 2014)<br>(Pan et al. 2020) |
| Inertie des murs | 0.70 × valeur initiale | 1.30 × valeur initiale | (Teughels, Maeck et De Roeck 2002) |

Finalement, la correspondance entre les formes modales numériques et expérimentales est évaluée par le critère CAM. Pour considérer qu'une forme modale est correctement calibrée, la valeur du CAM doit satisfaire : $CAM \geq 0.9$.

## 1.6 Conclusion de la revue

La revue de littérature présentée a permis de dresser un état des connaissances actuel sur l'identification modale, l'analyse de sensibilité et la calibration des modèles d'éléments finis appliquées aux bâtiments de grande hauteur. Les différentes méthodes d'extraction modale, qu'elles soient fréquentielles (DDFA) ou temporelles (ISS), permettent d'obtenir des données expérimentales fiables à partir de mesures in situ, nécessaires à la validation des modèles numériques. Parallèlement, les approches de sensibilité, notamment la méthode de Morris, se révèlent essentielles pour identifier les paramètres les plus influents à calibrer, tout en limitant le nombre de simulations requises.

Dans le cadre du présent mémoire, l'accent est mis sur une approche itérative de calibration physique du modèle, à partir de données modales mesurées sur une tour instrumentée. Les campagnes de mesures ont été planifiées conformément aux recommandations scientifiques récentes en matière de vibration ambiante : fréquence d'échantillonnage de 512 Hz, durée d'acquisition adéquate, position stratégique des capteurs autour du noyau de cisaillement et aux extrémités des dalles, un capteur placé au sommet de la structure, ainsi qu'une synchronisation radio systématique avant chaque essai.

L'analyse de sensibilité préalable a été réalisée à l'aide de la méthode de Morris, afin de restreindre le nombre de paramètres à calibrer tout en conservant ceux ayant le plus d'influence sur la réponse modale. Finalement, la calibration du modèle numérique repose sur l'algorithme de Svendsen (2020), illustré à la Figure 1.20, fondé sur la méthode des moindres carrés pondérés. Cette stratégie permet une mise à jour robuste et rigoureuse des propriétés physiques du modèle, tout en maintenant un bon compromis entre précision, stabilité numérique et faisabilité computationnelle.

# CHAPITRE 2

# ANALYSE MODALE BASÉE SUR LES VIBRATIONS AMBIANTES D'UN BÂTIMENT DE GRANDE HAUTEUR À MONTRÉAL

## 2.1 Contexte

Ce chapitre présente les mesures de vibrations ambiantes (MVA) effectuées sur une tour de grande hauteur ainsi que leur traitement. Il propose une analyse modale expérimentale visant à caractériser les paramètres dynamiques de la structure, notamment l'amortissement, les fréquences naturelles et les déformées modales. Une attention particulière est portée à l'amortissement, un paramètre difficile à quantifier pour les bâtiments de grande hauteur. Cette analyse constitue un préalable essentiel au chapitre 3, qui porte sur la calibration du modèle numérique à l'aide de ces données expérimentales.

## 2.2 Vibrations ambiantes

D'après le Gouvernement du Canada (2008), les vibrations ambiantes constituent un enjeu dans les bâtiments de grande hauteur. Elles proviennent de sources internes (machines, systèmes de ventilation CVC, ascenseurs, activités des occupants) et de sources externes (vent, chantiers voisins, trafic routier/ferroviaire). Ces sollicitations mettent en vibration les éléments horizontaux (dalles) et verticaux (murs et colonnes). Les bâtiments de grande hauteur sont notamment très sensibles au vent. Dans ce contexte, les MVA exploitent justement ces excitations naturelles pour identifier in situ les paramètres modaux (fréquences, amortissement, formes), étape clé pour la compréhension et la calibration des modèles numériques.

## 2.3 Méthodologie

Afin de caractériser la réponse modale de la tour à l'étude, une campagne de mesures in-situ a d'abord été réalisée. L'étape de prétraitement consiste à supprimer le segment temporel initial

Δt correspondant à la mise en place et à la stabilisation des vélocimètres. Cette opération est réalisée à l'aide de scripts développés sur Python, permettant d'importer les signaux temporels dans ARTeMIS afin d'y être traités. L'analyse modale, effectuée avec le logiciel ARTeMIS Modal Pro (SVIBS 2025), permet l'extraction des fréquences naturelles, des formes modales et de l'amortissement. La réalisation de mesures de vibrations ambiantes requiert l'utilisation d'accéléromètres ou de vélocimètres, capables d'enregistrer les vitesses de vibration au niveau des planchers. Dans le cadre de cette étude, cinq vélocimètres de type TROMINO® ont été utilisés, comme illustré à la Figure 2.1.


Figure 2.1 Les appareils TROMINO® utilisés dans cette étude

Chaque appareil est muni de sept canaux : trois vélocimètres et trois accéléromètres couvrant les trois directions de l'espace, ainsi qu'un canal analogique supplémentaire. La synchronisation entre les unités est assurée par transmission radio ou GPS, garantissant une cohérence temporelle entre les signaux. Ces appareils peuvent opérer dans une large bande passante allant de 0,1 à 1024 Hz, avec un bruit intrinsèque inférieur à 0,5 microvolt (valeur efficace) à 128 Hz, ce qui témoigne de la qualité et de la fiabilité des mesures obtenues (MoHo 2025).

Les vitesses de vibration sont mesurées à chaque degré de liberté, ce qui permet d'identifier les fréquences naturelles, d'estimer les amortissements et de reconstituer les déformées modales dans les directions de l'espace souhaitées. Dans cette étude, seules les directions X et Y sont analysées, la direction Z étant négligée en raison de la prédominance des effets gravitaires, qui limitent les déplacements modaux verticaux. Conformément aux recommandations présentées à la section 1.2 de la revue de littérature, un capteur de référence en l'occurrence, l'appareil vert visible sur la  Figure 2.1, est installé au point le plus élevé accessible de la structure, soit au 58e étage. Ce capteur reste fixe tout au long de la campagne d'essai et constitue la base de référence temporelle pour tous les enregistrements. Deux configurations de test sont réalisées à chaque étage. La première consiste à disposer les capteurs autour du système de résistance latérale (noyau de cisaillement ou cage d'ascenseur) afin de capter les composantes de translation des modes (nom du test : Trans. 1). La seconde configuration place les capteurs aux extrémités du bâtiment pour maximiser la détection des effets de torsion, comme recommandé à la section 1.2 (nom du test : Tors. 1). La disposition typique des capteurs par zone est illustrée à la Figure 2.2.

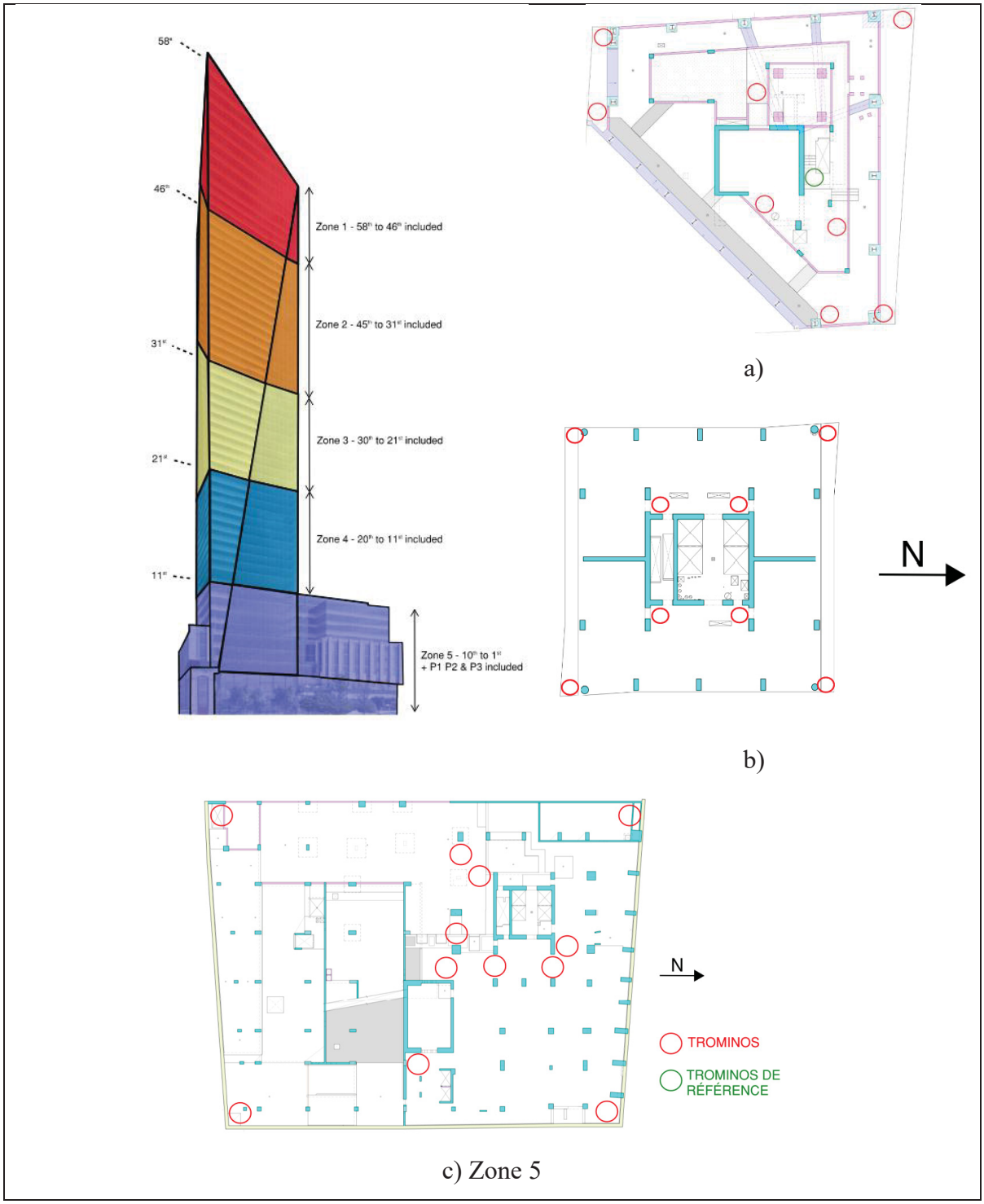


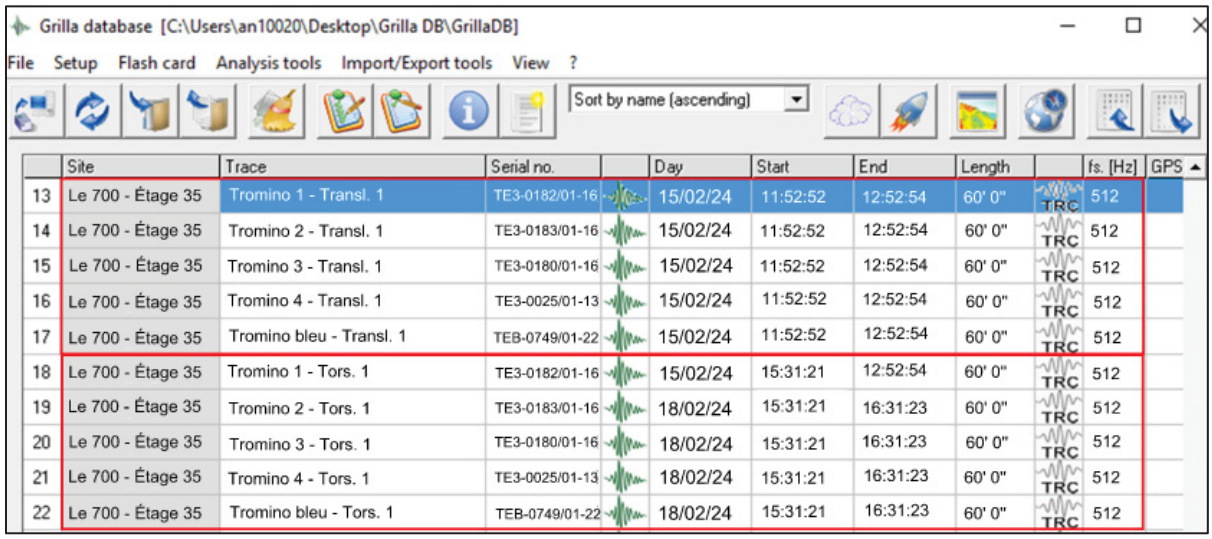

Figure 2.2 Position des appareils selon les zones du bâtiment : a) le dernier étage accessible, b) les zones 2,3, 4 et c) la zone 5

À noter qu'un troisième test (nom du test : Trans. 2) a été réalisé entre le 11$^{e}$ étage et le rez-de-chaussée, au niveau du podium, afin d'évaluer l'effet de couplage dynamique entre la tour et le podium, correspondant à la zone d'interface entre les deux structures.

La fréquence fondamentale est estimée à 0.23 Hz. Conformément aux recommandations formulées à la section 1.2, plusieurs paramètres de mesure ont été retenus pour garantir la qualité des données acquises. La fréquence d'échantillonnage a été fixée à 512 Hz, ce qui permet de capter les basses fréquences caractéristiques du comportement modal des bâtiments de grande hauteur tout en assurant une bande passante suffisante. La durée des enregistrements a été limitée à une heure par test, car une contrainte est imposée par les conditions d'accès in situ. Bien que cette durée soit inférieure à la durée minimale idéale de 3000 fois la période naturelle (soit 3000 x 1/0.23 = 3.62 heures) conformément à la section 1.2, elle demeure suffisante pour permettre une identification temporelle des paramètres modaux via les méthodes SSI (Stochastic Subspace Identification). Toutefois, cette limitation peut nuire à la précision des méthodes fréquentielles telles que la DDFA, notamment pour l'estimation de l'amortissement, qui nécessite généralement une plus longue durée d'enregistrement pour une stabilité accrue comme expliqué à la section 1.3.2 de la revue de littérature.

La résolution fréquentielle obtenue est de $2{,}78\times10^{-4}$ Hz, calculée à partir du ratio entre la fréquence d'échantillonnage et le nombre total de points NTFR, soit 512 Hz divisés par $1{,}84\times10^{6}$ points. Ce dernier correspond à la durée totale de mesure (3600 secondes) divisée par l'intervalle temporel entre deux échantillons (1/512 s). Enfin, afin d'assurer une cohérence temporelle parfaite entre les différents capteurs lors des mesures, une synchronisation est effectuée avant chaque essai à l'aide d'un système de transmission radio. Cette synchronisation garantit un horodatage précis des signaux, indispensable pour les méthodes d'analyse reposant sur l'exploitation simultanée des signaux mesurés par plusieurs capteurs, notamment pour l'identification des déformées modales et l'estimation de l'amortissement à partir de la cohérence entre capteurs situés à différents étages.

Un fois les mesures terminées, les appareils sont connectés à un ordinateur afin de récupérer leurs enregistrements via le logiciel GRILLA®. Toutes les traces (soit les signaux enregistrés par les appareils) sont répertoriées dans la base de données. Un exemple de celle-ci est montré sur la Figure 2.3. Ce logiciel permet de synchroniser les signaux de chaque capteur, enregistrés via transmission radio, et garantit que la trace globale commence et se termine au même instant de l'horloge interne de chaque capteur.




| | Site | Trace | Serial no. | Day | Start | End | Length | fs. [Hz] | GPS |
|---|---|---|---|---|---|---|---|---|---|
| 13 | Le 700 - Étage 35 | Tromino 1 - Transl. 1 | TE3-0182/01-16 | 15/02/24 | 11:52:52 | 12:52:54 | 60' 0" | 512 | |
| 14 | Le 700 - Étage 35 | Tromino 2 - Transl. 1 | TE3-0183/01-16 | 15/02/24 | 11:52:52 | 12:52:54 | 60' 0" | 512 | |
| 15 | Le 700 - Étage 35 | Tromino 3 - Transl. 1 | TE3-0180/01-16 | 15/02/24 | 11:52:52 | 12:52:54 | 60' 0" | 512 | |
| 16 | Le 700 - Étage 35 | Tromino 4 - Transl. 1 | TE3-0025/01-13 | 15/02/24 | 11:52:52 | 12:52:54 | 60' 0" | 512 | |
| 17 | Le 700 - Étage 35 | Tromino bleu - Transl. 1 | TEB-0749/01-22 | 15/02/24 | 11:52:52 | 12:52:54 | 60' 0" | 512 | |
| 18 | Le 700 - Étage 35 | Tromino 1 - Tors. 1 | TE3-0182/01-16 | 15/02/24 | 15:31:21 | 12:52:54 | 60' 0" | 512 | |
| 19 | Le 700 - Étage 35 | Tromino 2 - Tors. 1 | TE3-0183/01-16 | 18/02/24 | 15:31:21 | 16:31:23 | 60' 0" | 512 | |
| 20 | Le 700 - Étage 35 | Tromino 3 - Tors. 1 | TE3-0180/01-16 | 18/02/24 | 15:31:21 | 16:31:23 | 60' 0" | 512 | |
| 21 | Le 700 - Étage 35 | Tromino 4 - Tors. 1 | TE3-0025/01-13 | 18/02/24 | 15:31:21 | 16:31:23 | 60' 0" | 512 | |
| 22 | Le 700 - Étage 35 | Tromino bleu - Tors. 1 | TEB-0749/01-22 | 18/02/24 | 15:31:21 | 16:31:23 | 60' 0" | 512 | |

Figure 2.3 Exemple de la base de données GRILLA®

Les encadrés rouges correspondent à un test composé de 5 appareils. Ainsi, avant d'exporter les données pour pré-traitement dans un script Python®, il faut s'assurer de lier les traces de chaque test en utilisant l'option « synchronisation par radio » dans l'onglet « analysis tools ». Puis avec l'outil « export tool », les signaux sont exportés en cochant l'option « natural frequencies ».

Par la suite, un script Python est utilisé pour exporter les données dans un format compatible avec le logiciel ARTeMIS Modal Pro. Le script supprime aussi le décalage temporel $\Delta t$ entre le début d'acquisition et l'instant où tous les capteurs sont en place et stabilisés**.** En effet, certains enregistrements démarrent alors qu'un capteur est encore en déplacement ou en voie d'installation. En début de test, le segment initial non représentatif est donc tronqué pour ne

conserver que les données valides. À titre d'exemple, la Figure 2.4 illustre les signaux enregistrés au 35e étage avant (Figure 2.4a) et après (Figure 2.4b) traitement, dans les directions nord-sud (NS) et ouest-est (OE).

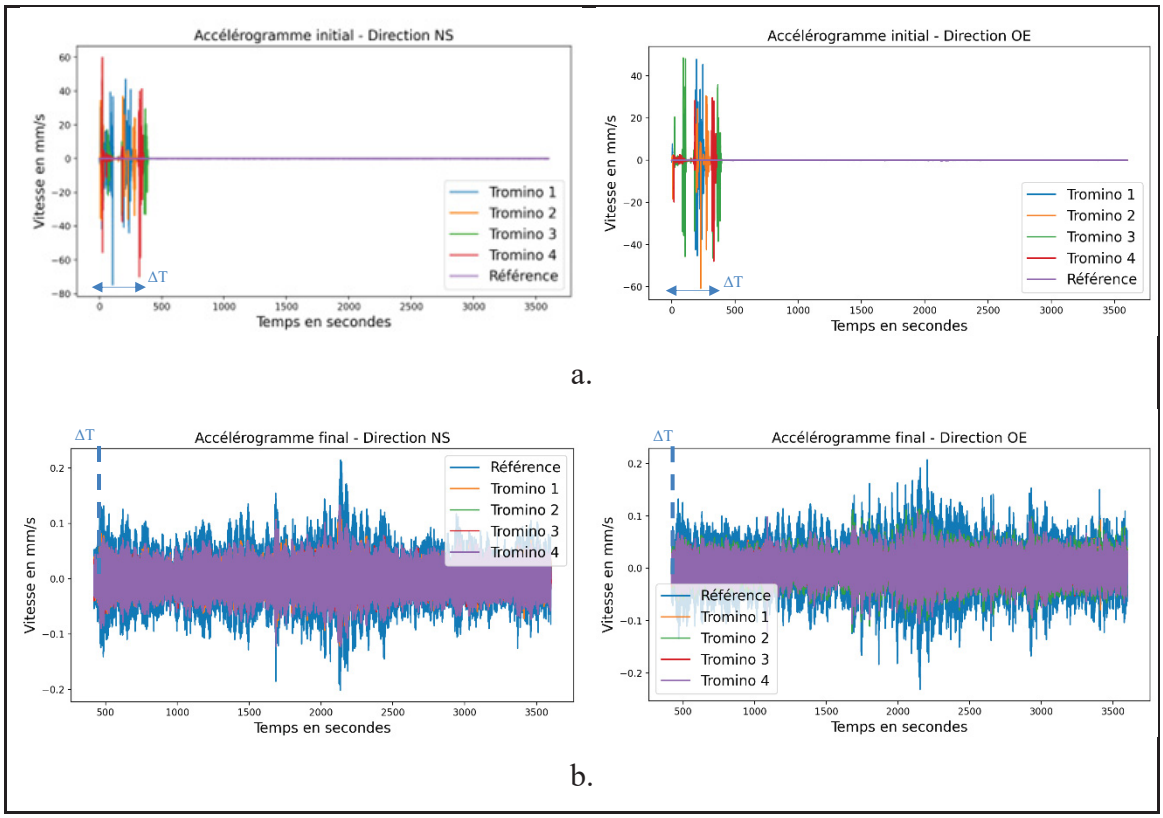



Figure 2.4 Accélérogramme de l'étage 35 a) avec et b) sans Δt dans les directions NS et EO

Ici, le Δt supprimé correspond à l'intervalle 0–500 secondes où des variations de vitesse importantes sont observées de l'ordre de ± 60 mm/s sur ce segment. Une fois ce Δt enlevé, les signaux de vitesse sont compris entre 0 et plus ou moins 0.2 mm/s, qui sont des valeurs plus logiques. Enfin, compte tenu du nombre d'échantillons élevé (NTFR important) et de la résolution fréquentielle fine (Δf faible), l'échantillonnage est jugé suffisant. Les signaux sont donc importés tels quels dans ARTeMIS Modal Pro.

## 2.4 Traitement des signaux en utilisant le logiciel ARTeMIS Modal Pro

Une fois l'étape de pré-traitement réalisée avec un script Python, les données sont importées dans le logiciel ARTeMIS Modal Pro (SVIBS 2025) pour évaluer les paramètres modaux. Les deux méthodes temporelles ISS-COV (soit SSI-UPC sur ARTeMIS) et SSI-DONNÉE (SSI-UP sur ARTeMIS) sont utilisées ainsi que la méthode fréquentielle DDFA pour limiter les biais.

### 2.4.1 Préparation des données : tendance parasite, décimation et filtration

La première étape est de préparer les données mesurées sur site pour l'analyse modale temporelle (ISSs) ou fréquentielle (DDFA). Certaines options sont choisies dans l'interface (Figure 2.5) de traitement des signaux.

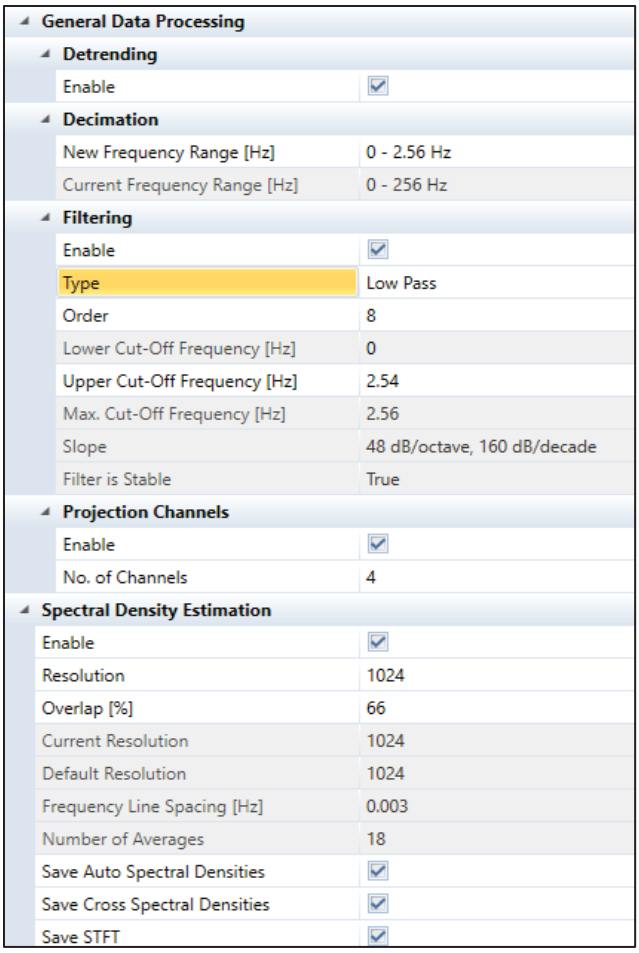



Figure 2.5 Interface de traitement des signaux d'ARTeMIS Modal Pro (SVIBS 2025)

La première option consiste à supprimer les tendances parasites (« detrending » en anglais), dues notamment aux dérives instrumentales (décalage dans le temps des appareils) ou des effets de température (variation lente des capteurs). Activer cette option est essentiel pour éviter l'ajout de basses fréquences non physiques dans le spectre et garantir que l'analyse se concentre sur les vibrations modales.

L'option 2 permet de restreindre l'analyse à une plage de fréquences spécifique afin de réduire la quantité de données à traiter et de se concentrer sur les composantes fréquentielles pertinentes. Dans ce mémoire, cette plage est comprise entre 0 et 2.56 Hz, choix justifié par le fait que les bâtiments de grande hauteur présentent généralement leurs premiers modes de vibration dans cet intervalle. Les signaux mesurés sont ensuite traités à l'aide de l'option

« Filtering », en appliquant un filtre passe-bas d'ordre 8 avec une fréquence de coupure fixée à 2.56 Hz, tandis que les autres paramètres de cette section sont conservés à leurs valeurs par défaut. Ce filtrage permet de préserver les basses fréquences d'intérêt tout en atténuant progressivement les hautes fréquences, réduisant ainsi les artefacts numériques et garantissant une qualité d'analyse adéquate.

Les channels de projection permettent d'améliorer l'identification modale en projetant les signaux sur les directions optimales compensant les erreurs de placement/d'orientation des capteurs, séparant les modes proches et en améliorant le rapport signal/bruit (SVIBS 2025).

Finalement, la densité spectrale est estimée à l'aide d'une transformée de Fourier rapide de 1024 points avec un chevauchement de 66 %. L'utilisation d'un nombre de points plus élevé permettrait d'améliorer la résolution fréquentielle, mais au prix d'un temps de calcul plus important. Le choix de 1024 points constitue ainsi un compromis adéquat entre précision spectrale et efficacité computationnelle. Le chevauchement de 66 % correspond à la portion de la fenêtre de la FFT réutilisée pour le calcul de la fenêtre suivante, ce qui permet de lisser le spectre et d'en améliorer la stabilité tout en maintenant un temps de calcul raisonnable.

### 2.4.2 Suppression des fréquences harmoniques

Une harmonique est une fréquence multiple d'une fréquence fondamentale générée par une source d'excitation périodique (machines rotatives, réseau électrique, trains, véhicules…). Ainsi, lors de l'évaluation des paramètres modaux et notamment des fréquences naturelles du bâtiment, il faut s'assurer que les fréquences harmoniques soient supprimées et différentiées des fréquences naturelles pour ne pas fausser l'analyse des modes naturels de vibrations de la structure avec des fréquences parasites.

Sur ARTeMIS Modal Pro, l'algorithme de détection d'harmoniques génère la fréquence moyenne Kurtosis de tous les canaux (X, Y et Z : axes dans l'espace) de mesures projetées. La kurtosis est un indicateur statistique décrivant la forme de la distribution d'un signal, sensible

à la présence de pics ou de composantes périodiques marquées. En analyse modale opérationnelle, des valeurs élevées de kurtosis sont généralement associées à des composantes harmoniques, tandis que les modes propres physiques présentent des valeurs plus modérées.

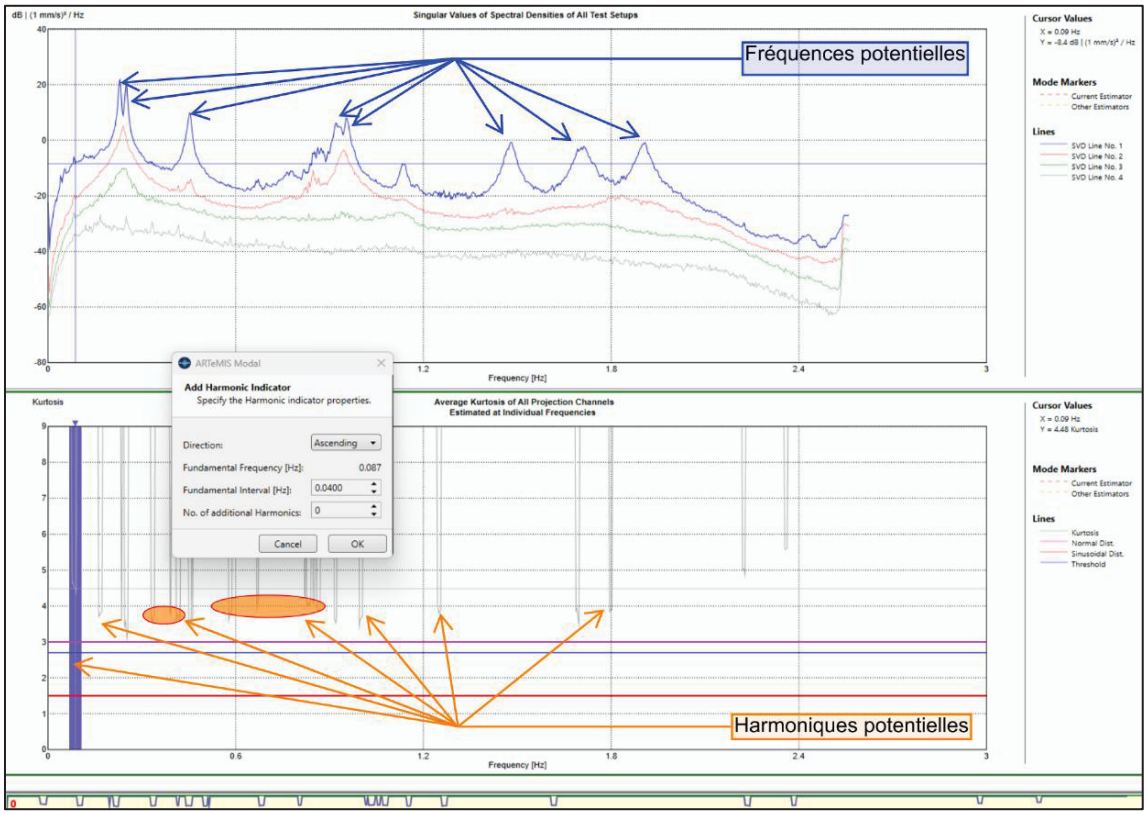



Figure 2.6 Graphique des valeurs singulières et de détection d'harmoniques

Grâce au graphique des valeurs singulières (Figure 2.6), il est possible d'identifier les fréquences potentielles correspondant aux pics observés et les pics récurrents pouvant correspondre à des harmoniques qui se répètent. À noter que le pic autour de 1.1 Hz n'est pas considéré comme une fréquence naturelle. Les pics se sélectionnent avec l'outil « Add Harmonic Indicator ». La bande doit couvrir les pics de la détection d'harmoniques, qui se répètent périodiquement, avec l'option « No. of additional Harmonics ».

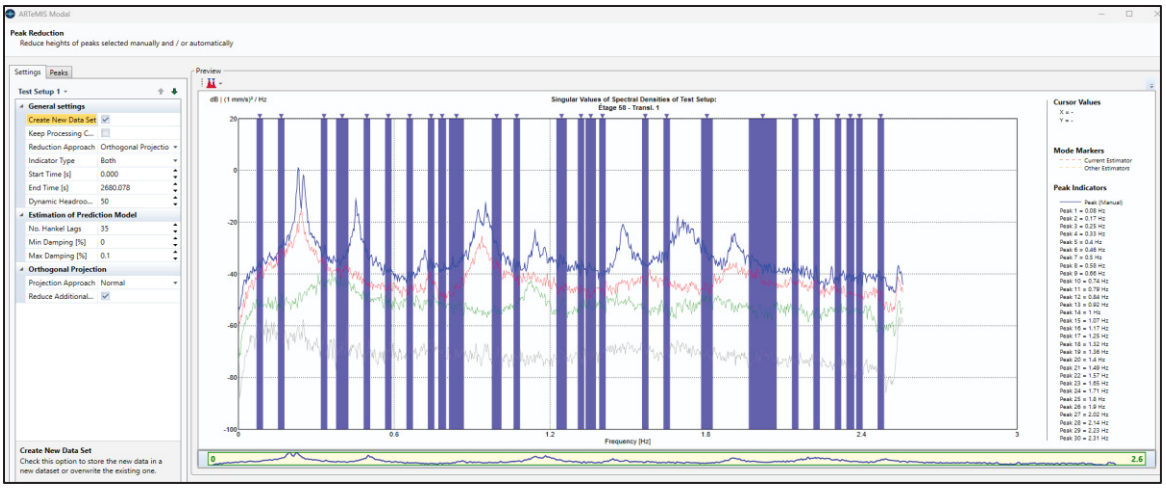



Figure 2.7 Interface de réduction des harmoniques

Une fois sélectionnés les pics d'harmoniques (exemple : bandes bleues sur la Figure 2.7), il est possible de réduire, et pour chaque test, ces pics comme montré sur la Figure 2.7. Ceci permet donc de créer de nouvelles données filtrées pour chaque test et de se concentrer uniquement sur l'analyse des pics de fréquences naturelles et non des pics parasites.

### 2.4.3 Normalisation des formes modales

À la suite de l'analyse, les déformées modales exportée d'ARTeMIS Modal Pro sont de forme complexe. Or, d'après Chopra (2014), il est de bonne pratique de garder la partie réelle puisque les parties complexes sont négligeable et très petites. Les données exportées d'ARTeMIS sont montrées par la Figure 2.8, ci-dessous.

```
Structural Vibration Solutions A/S - www.svibs.com

SVS Configuration File written for PC system
File was generated by ARTeMIS Modal

BEGIN MODE DEFINITION

PROJECT
        C:\Users\cyrie\OneDrive\Documents\Mémoire\Le 700 St Jacques\Modèle ARTeMIS\Modèle.amp

ESTIMATOR
        SSI-PC

CREATION: DATE / TIME
        12-12-2024  18:48:48

FREQUENCY [HZ]: MEAN / SDEV
        2.295449e-01    1.195961e-03

DAMPING [%]: MEAN / SDEV
        7.660094e-01    6.781323e-01

MODE SHAPE: NODE / X-ABS / X-ANG / Y-ABS / Y-ANG / Z-ABS / Z-ANG
     1  3.637609e-03 -1.414240e-01      1.658719e-03  4.003737e-01      0.000000e+00  0.000000e+00
     2  4.648065e-03 -2.992115e+00      3.657291e-02  3.100789e+00      0.000000e+00  0.000000e+00
     3  0.000000e+00  0.000000e+00      0.000000e+00  0.000000e+00      0.000000e+00  0.000000e+00
     4  1.802301e-02 -3.819511e-02      1.572878e-02 -3.137081e+00      0.000000e+00  0.000000e+00
     5  1.156820e-02  3.093051e+00      2.655318e-02 -3.139757e+00      0.000000e+00  0.000000e+00
     6  8.494212e-02  3.125129e+00      2.481603e-02  3.045326e+00      0.000000e+00  0.000000e+00
     7  8.922710e-02  3.123371e+00      4.059506e-01 -3.139361e+00      0.000000e+00  0.000000e+00
     8  5.497688e-01 -3.132155e+00      4.190878e-01 -3.131954e+00      0.000000e+00  0.000000e+00
     9  4.548774e-01  8.585182e-03      4.750545e-02  3.106481e+00      0.000000e+00  0.000000e+00
    10  4.054052e-01  2.878070e+00      8.069682e-02  3.138374e+00      0.000000e+00  0.000000e+00
    11  2.651174e-01 -3.067020e+00      8.525369e-01 -3.097337e+00      0.000000e+00  0.000000e+00
    12  2.447881e-01 -2.901269e+00      8.836603e-01 -3.112264e+00      0.000000e+00  0.000000e+00
    13  3.086866e-01 -8.199716e-01      7.311350e-01  3.075613e+00      0.000000e+00  0.000000e+00
    14  5.151238e-01 -4.226914e-02      6.077417e-01 -3.130325e+00      0.000000e+00  0.000000e+00
```

Figure 2.8 Données des formes modales exportées du logiciel ARTeMIS Modal Pro

Ces données donnent la magnitude en X et Y (X-ABS, Y-ABS), et les angles de phase des coordonnées X et Y (X-ANG, Y-ANG) permettant notamment de déterminer les parties réelles de chaque coordonnée pour chaque direction. Pour ce faire, la magnitude en X ou Y est multiplié par le cosinus de l'angle de la coordonnée étudiée, c-à-d X ou Y. Si la partie imaginaire est recherchée, la même méthode est appliquée en appliquant le sinus et non le cosinus. À noter que les composantes du vecteur de la déformée modale combinent les coordonnées X et Y dans un unique vecteur.

Un fois la partie réelle obtenue, on normalise le vecteur $\varphi$ selon la méthodologie fournie par ARTeMIS Modal Pro, (SVIBS 2025) :

$$\widetilde{\varphi}_\iota = \frac{\varphi_i}{\varphi_{i,k}}$$
$$\breve{\varphi}_\iota = \frac{\widetilde{\varphi}_\iota}{\|\widetilde{\varphi}_\iota\|} \tag{2.1}$$

Cette normalisation utilisée par ARTeMIS est faite en deux étapes. La première consiste à prendre les valeurs maximales de la déformée modale à un DOF précis en X puis Y, et à diviser toutes les autres valeurs de déformées modales avec celles-ci. On obtient donc un autre vecteur $\widetilde{\varphi}_\iota$ ayant pour valeur maximale 1. En général, la valeur maximale se trouve souvent au sommet du bâtiment.

La seconde étape de normalisation consiste à reprendre ce vecteur $\widetilde{\varphi}_\iota$ et le diviser par sa norme euclidienne pour obtenir $\breve{\varphi}_\iota$. D'autres normalisations existent telles que la normalisation par la masse. Cette dernière sera utilisée dans le CHAPITRE 3 de ce mémoire lors de la calibration du MÉF.

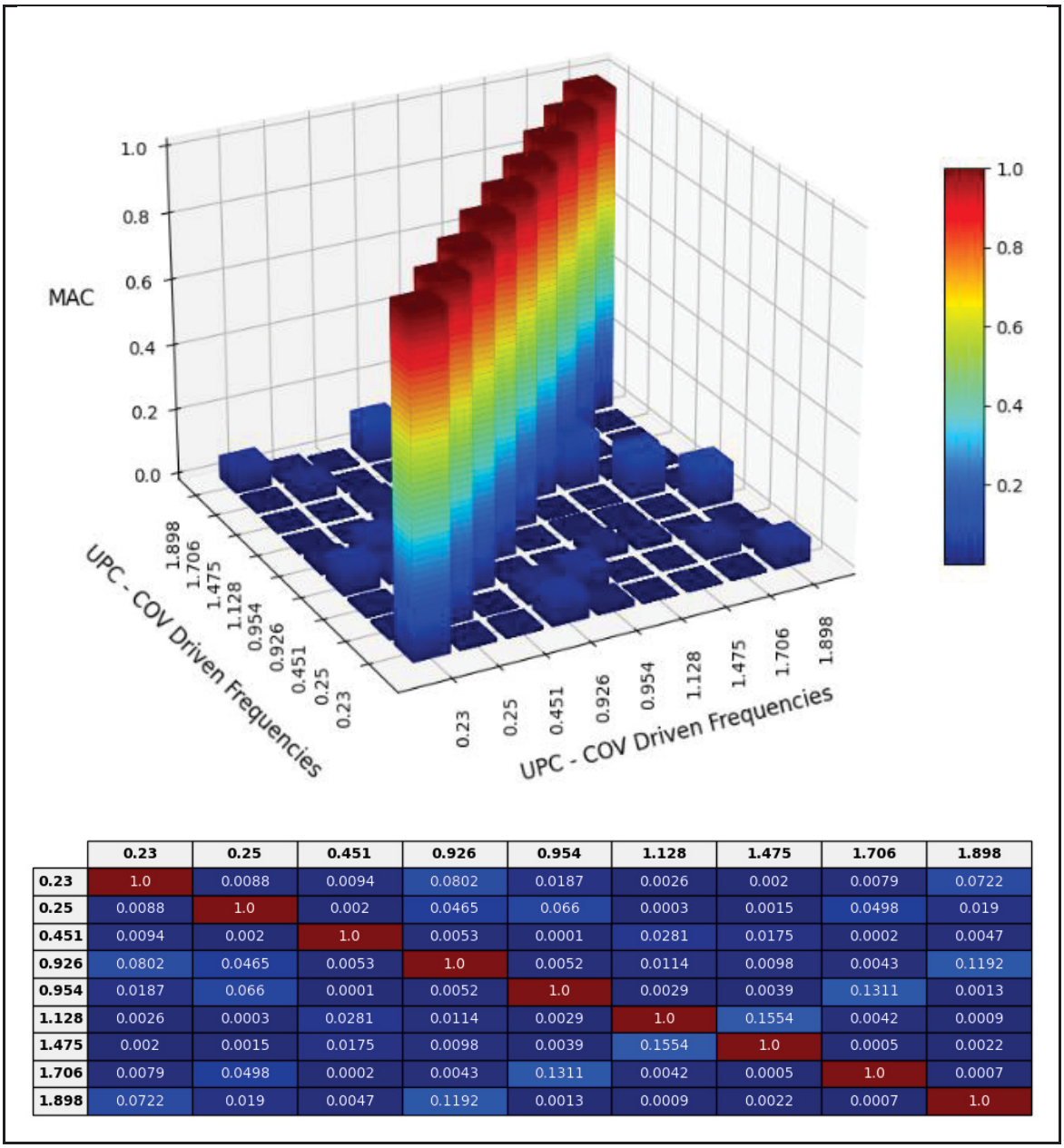
## 2.5 Résultats expérimentaux

La Figure 2.9 montre une excellente cohérence au niveau des neuf premiers modes extraits, validant la robustesse des résultats malgré la présence d'harmoniques dans les mesures. En effet, la Figure 2.11b montre des fréquences stables quel que soit la méthode d'extraction regardée.


Figure 2.9 Résultats expérimentaux de l'analyse modale : a) Amortissement (en %) et b) Fréquence (Hz) en fonction des modes

Cependant, une divergence de résultats concernant l'amortissement est observée pour les modes supérieurs à 4 (Figure 2.11a). La méthode DDFA a tendance à sous-estimer significativement les amortissements par rapport aux méthodes ISSs. Surtout pour les modes 4, 5, 6 et 9, cette différence peut s'expliquer par la forte influence des harmoniques dans ces bandes de fréquence, rendant l'identification du pic de résonance plus difficile pour DDFA,

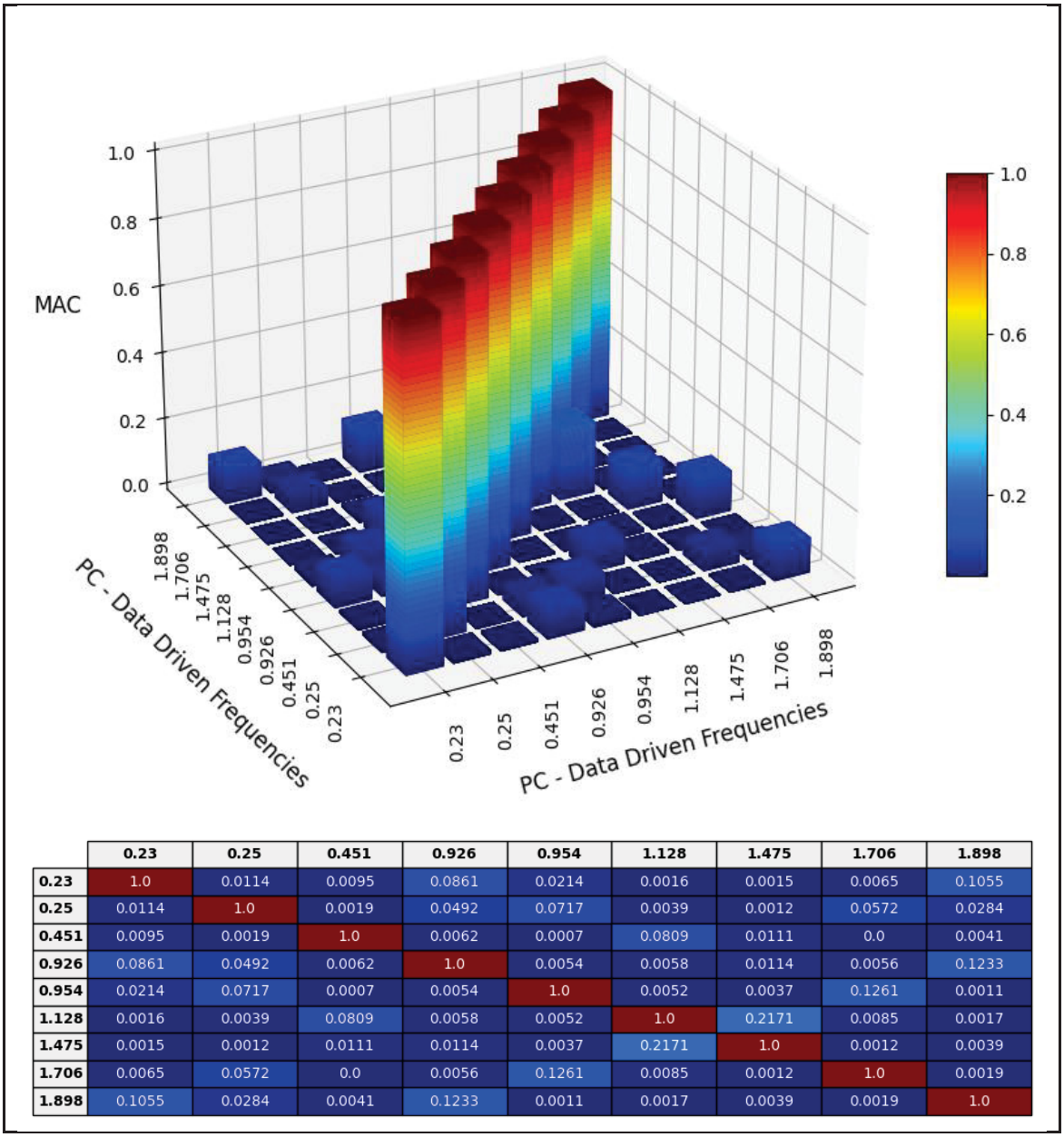
qui repose sur l'extraction spectrale et le contenu fréquentiel. Cette observation souligne l'intérêt des méthodes ISSs basées sur le domaine temporel, qui sont mieux adaptées aux environnements vibratoires perturbés comme les bâtiments de grande hauteur. Ainsi, pour la suite de ce mémoire, les méthodes temporelles ISSs sont utilisées.




| | 0.23 | 0.25 | 0.451 | 0.926 | 0.954 | 1.128 | 1.475 | 1.706 | 1.898 |
|---|---|---|---|---|---|---|---|---|---|
| **0.23** | 1.0 | 0.0088 | 0.0094 | 0.0802 | 0.0187 | 0.0026 | 0.002 | 0.0079 | 0.0722 |
| **0.25** | 0.0088 | 1.0 | 0.002 | 0.0465 | 0.066 | 0.0003 | 0.0015 | 0.0498 | 0.019 |
| **0.451** | 0.0094 | 0.002 | 1.0 | 0.0053 | 0.0001 | 0.0281 | 0.0175 | 0.0002 | 0.0047 |
| **0.926** | 0.0802 | 0.0465 | 0.0053 | 1.0 | 0.0052 | 0.0114 | 0.0098 | 0.0043 | 0.1192 |
| **0.954** | 0.0187 | 0.066 | 0.0001 | 0.0052 | 1.0 | 0.0029 | 0.0039 | 0.1311 | 0.0013 |
| **1.128** | 0.0026 | 0.0003 | 0.0281 | 0.0114 | 0.0029 | 1.0 | 0.1554 | 0.0042 | 0.0009 |
| **1.475** | 0.002 | 0.0015 | 0.0175 | 0.0098 | 0.0039 | 0.1554 | 1.0 | 0.0005 | 0.0022 |
| **1.706** | 0.0079 | 0.0498 | 0.0002 | 0.0043 | 0.1311 | 0.0042 | 0.0005 | 1.0 | 0.0007 |
| **1.898** | 0.0722 | 0.019 | 0.0047 | 0.1192 | 0.0013 | 0.0009 | 0.0022 | 0.0007 | 1.0 |

Figure 2.10 Critère d'assurance modal CAM pour la méthode temporelle ISS-COV (SSI-COV ou SSI-UPC sur ARTeMIS)

Les Figure 2.10 et Figure 2.11 présentent les matrices CAM obtenues respectivement pour les méthodes temporelles ISS-COV (ou SSI-UPC sur ARTeMIS) et ISS-DONNÉE (ou SSI-PC sur ARTeMIS). Dans les deux cas, les valeurs élevées observées sur la diagonale principale, proches de 1.0, indiquent une excellente correspondance modale entre les fréquences identifiées, ce qui témoigne d'une extraction robuste et fiable des modes propres. Ces valeurs élevées sur la diagonale sont le signe d'une forte cohérence entre les vecteurs propres obtenus à chaque fréquence, caractéristique attendue pour une structure stable excitée par des vibrations ambiantes.




| | 0.23 | 0.25 | 0.451 | 0.926 | 0.954 | 1.128 | 1.475 | 1.706 | 1.898 |
|---|---|---|---|---|---|---|---|---|---|
| **0.23** | 1.0 | 0.0114 | 0.0095 | 0.0861 | 0.0214 | 0.0016 | 0.0015 | 0.0065 | 0.1055 |
| **0.25** | 0.0114 | 1.0 | 0.0019 | 0.0492 | 0.0717 | 0.0039 | 0.0012 | 0.0572 | 0.0284 |
| **0.451** | 0.0095 | 0.0019 | 1.0 | 0.0062 | 0.0007 | 0.0809 | 0.0111 | 0.0 | 0.0041 |
| **0.926** | 0.0861 | 0.0492 | 0.0062 | 1.0 | 0.0054 | 0.0058 | 0.0114 | 0.0056 | 0.1233 |
| **0.954** | 0.0214 | 0.0717 | 0.0007 | 0.0054 | 1.0 | 0.0052 | 0.0037 | 0.1261 | 0.0011 |
| **1.128** | 0.0016 | 0.0039 | 0.0809 | 0.0058 | 0.0052 | 1.0 | 0.2171 | 0.0085 | 0.0017 |
| **1.475** | 0.0015 | 0.0012 | 0.0111 | 0.0114 | 0.0037 | 0.2171 | 1.0 | 0.0012 | 0.0039 |
| **1.706** | 0.0065 | 0.0572 | 0.0 | 0.0056 | 0.1261 | 0.0085 | 0.0012 | 1.0 | 0.0019 |
| **1.898** | 0.1055 | 0.0284 | 0.0041 | 0.1233 | 0.0011 | 0.0017 | 0.0039 | 0.0019 | 1.0 |

Figure 2.11 Critère d'assurance modal CAM pour la méthode temporelle ISS-DONNÉE (SSI-Data ou SSI-PC sur ARTeMIS)

Cependant, on remarque également que la méthode ISS-COV présente une meilleure séparation modale, notamment au niveau des valeurs hors-diagonale qui restent proches de zéro. Cela suggère une distinction plus marquée entre les différents modes, réduisant les risques de couplage modal ou de contamination entre modes adjacents. À titre d'exemple, la fréquence identifiée à 1.475 Hz dans la méthode ISS-COV présente une très faible interférence

avec les autres modes, contrairement à la méthode ISS-DONNÉE où l'on note quelques valeurs CAM hors-diagonale plus élevées (par exemple entre 1.128 Hz et 1.475 Hz), traduisant une séparation modale légèrement moins nette. Cette différence est cohérente avec les observations de la littérature : la méthode dirigée par la covariance (ISS-COV) est généralement plus robuste au bruit, et permet une meilleure distinction entre modes, particulièrement utile pour les structures de grande hauteur soumises à des niveaux d'excitation très faibles.

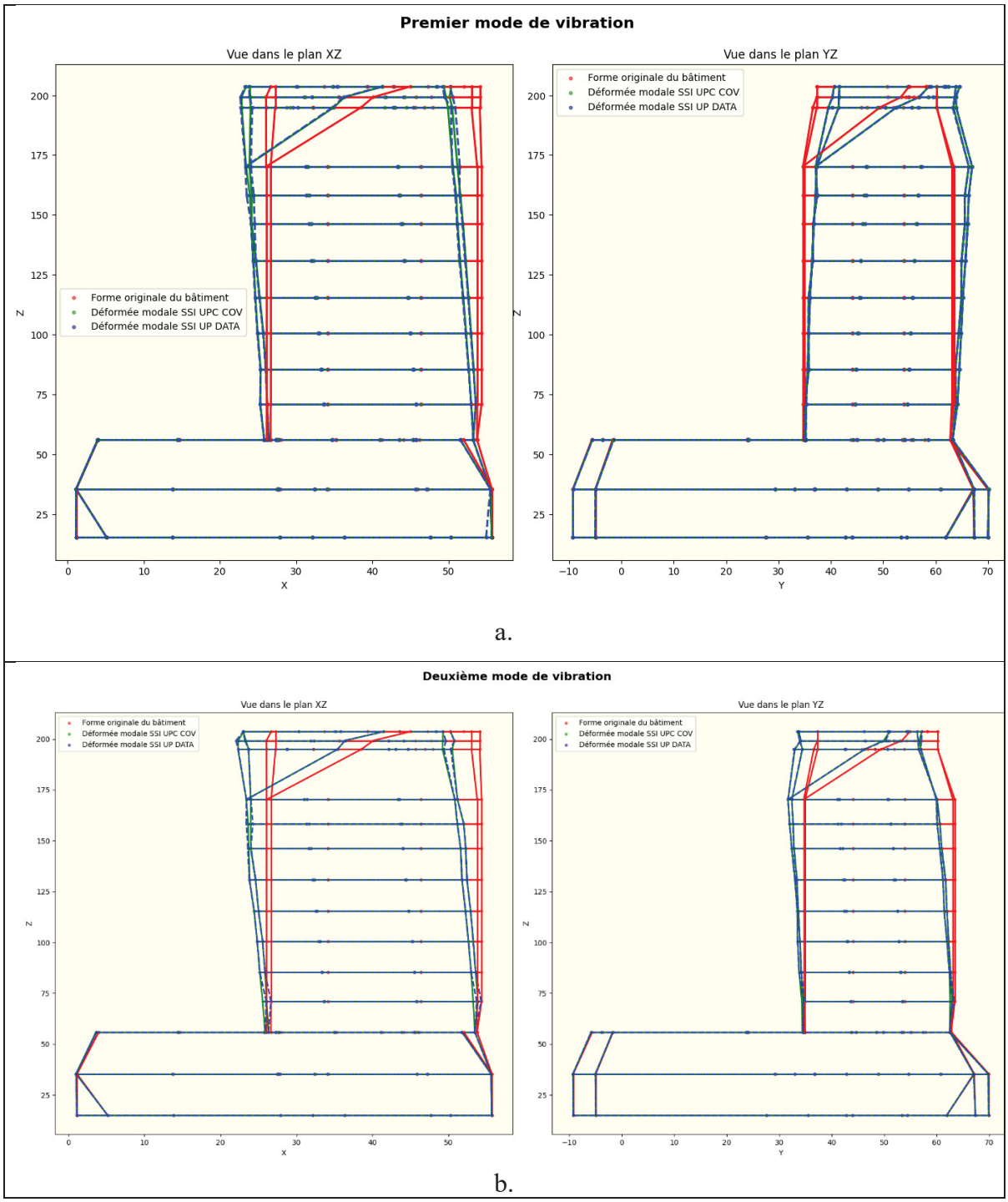



Figure 2.12 Déformées modales expérimentales du a) 1$^{er}$ mode et b) du 2$^{ème}$ mode

Les Figure 2.12, Figure 2.13 et Figure 2.14 représentant les cinq premiers modes de vibration permettent d'observer visuellement les déformées modales estimées à l'aide des deux variantes

de la méthode ISSs. Dans l'ensemble, on constate une excellente correspondance entre les deux méthodes, les formes modales superposées présentant une cohérence très élevée à travers les étages. Les différences les plus notables apparaissent aux niveaux supérieurs, où les amplitudes sont plus importantes et où les effets de torsion et d'interaction entre les modes deviennent plus sensibles.

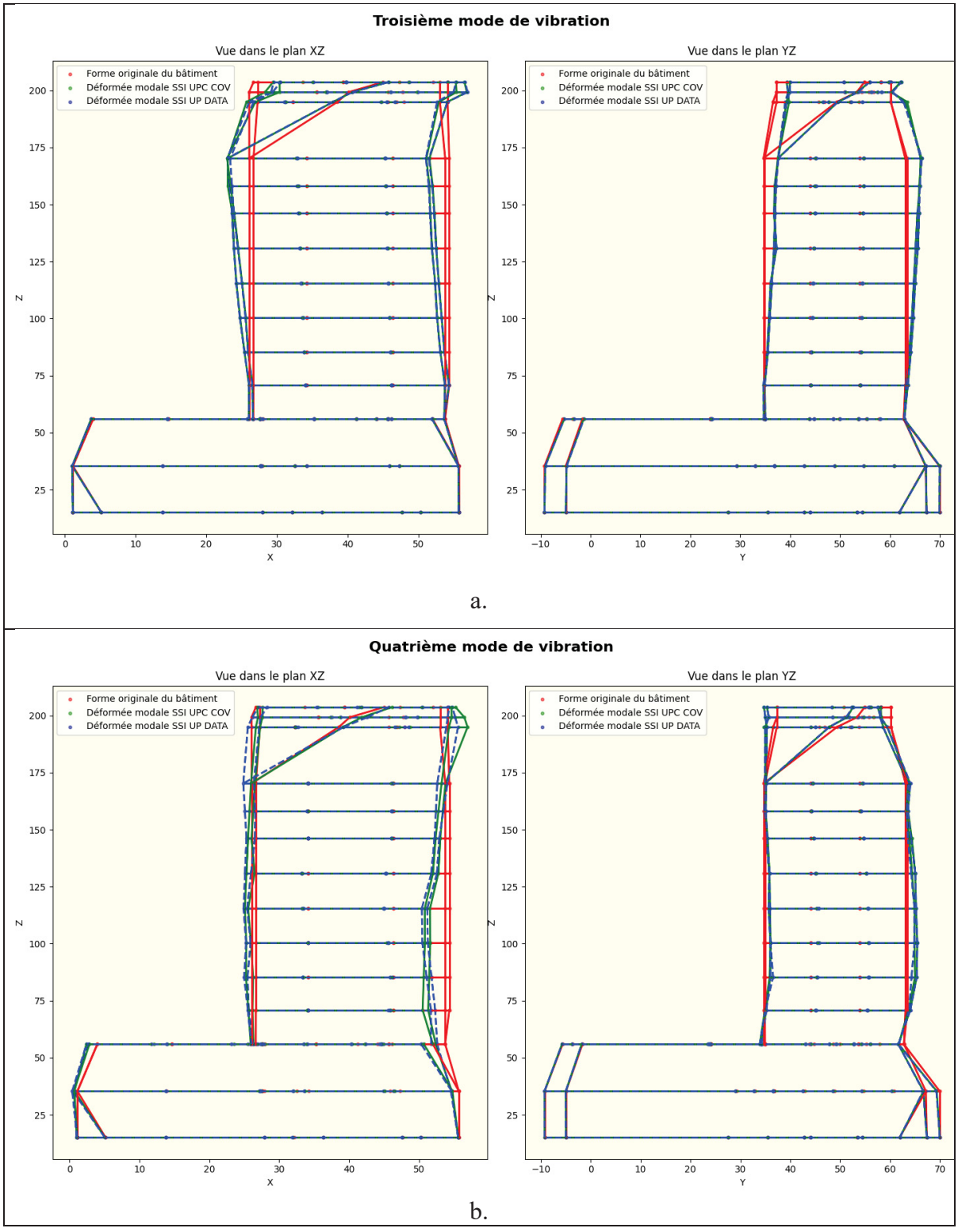



Figure 2.13 Déformées modales expérimentales du a) 3ème mode et b) du 4ème mode

Sur cet ensemble de Figure 2.12Figure 2.13 et Figure 2.14, la méthode ISS-COV (en vert) présente généralement des courbes plus lisses et moins bruitées, notamment aux extrémités, ce qui est cohérent avec sa robustesse accrue face aux faibles niveaux d'excitation. À l'inverse, la méthode ISS-DONNÉE (en bleu) montre parfois de légers écarts dans les zones les plus sollicitées, particulièrement visibles sur les déformées latérales au sommet dans les plans XZ et YZ. Ces écarts peuvent refléter une plus grande sensibilité au bruit ambiant, ce qui est cohérent avec le contexte de mesure, les étages supérieurs étant encore en construction et le bâtiment globalement non fermé, le rendant plus vulnérable à l'action du vent s'engouffrant dans la structure et induisant des niveaux d'accélération plus élevés au niveau des planchers.

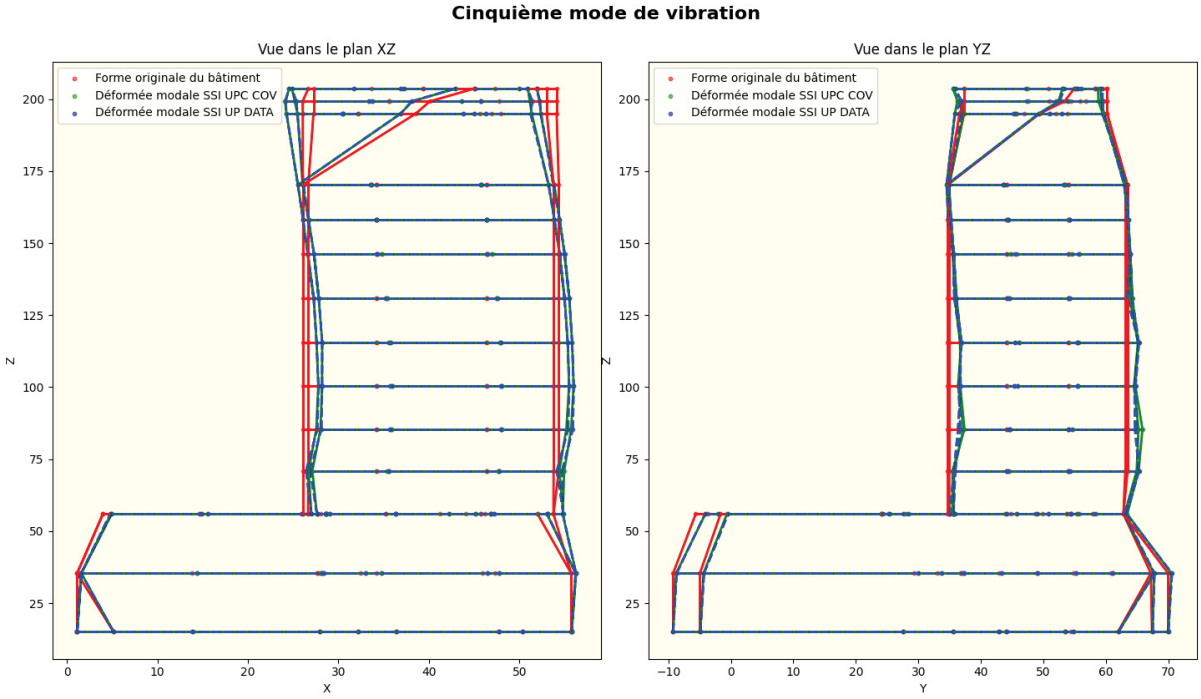



Figure 2.14 Déformées modales expérimentales du 5ème mode

La forme du quatrième mode (Figure 2.13b) est particulièrement révélatrice de ces nuances, avec des composantes en torsion plus marquées dans les résultats ISS-DONNÉE. Cela illustre la capacité de cette méthode à détecter les couplages modaux complexes, mais également sa sensibilité aux erreurs de phase ou de synchronisation. Enfin, la superposition avec la géométrie originale du bâtiment (en rouge) met en évidence des déplacements dominants dans

les directions X et Y, confirmant le caractère flexionnel des premiers modes. Dans l'ensemble, la cohérence entre les deux méthodes valide la fiabilité des résultats expérimentaux et démontre que les formes modales estimées sont représentatives du comportement réel de la structure instrumentée.

## 2.6 Recommandations

À la lumière des résultats obtenus, plusieurs recommandations peuvent être formulées pour les futures campagnes de mesures et analyses modales de bâtiments de grande hauteur. Tout d'abord, la méthode ISS-COV (SSI-COV ou SSI-UPC sur ARTeMIS) s'est montrée plus stable et précise que la méthode ISS-DONNÉE (SSI-Data ou SSI-PC sur ARTeMIS), en particulier dans les étages supérieurs où le signal est plus bruité. Les méthodes temporelles, comme les variantes SSI utilisées ici, se sont globalement avérées plus fiables que les méthodes fréquentielles pour l'extraction des amortissements et des formes modales, en particulier dans des conditions de faibles niveaux d'excitation, ce qui en fait des approches mieux adaptées aux mesures de vibrations ambiantes. D'un point de vue instrumental, la configuration testée dans cette étude, avec un capteur de référence permanent au sommet et des dispositions successives autour du noyau et aux extrémités, permet une capture efficace des modes translatifs et torsionnels, et devrait être privilégiée dans les futurs essais. Par ailleurs, les résultats suggèrent que l'utilisation d'un temps d'acquisition plus long permettrait une meilleure précision pour les méthodes fréquentielles comme la DDFA, notamment en ce qui concerne l'estimation de l'amortissement. De plus, une décimation supérieure à 2,54 Hz pourrait être nécessaire pour bien capturer les modes supérieurs ($6^{ème}$ mode et suivants), souvent cruciaux dans l'analyse dynamique des structures élancées. Enfin, l'amortissement mesuré in situ s'est révélé bien inférieur à la valeur normative de 2 % couramment utilisée au Canada. Dans le cas de la tour instrumentée, les valeurs observées étaient nettement inférieures à 1 %, ce qui remet en question la validité de cette hypothèse pour les bâtiments de grande hauteur. Une telle surestimation normative pourrait entraîner une sous-évaluation des effets dynamiques, en particulier pour les critères aux états limites de service (ÉLS) liés au confort des occupants.

Ce constat renforce l'importance de mener des études complémentaires sur d'autres tours afin de confirmer cette tendance. Il est également crucial, pour les firmes de génie-conseil québécoises, de disposer de valeurs locales et représentatives, plutôt que de s'appuyer sur des coefficients normatifs génériques établis à l'échelle nationale. Développer une base de données spécifique au contexte québécois permettrait de renforcer la fiabilité des modèles et des hypothèses employées en conception dynamique.

## CHAPITRE 3

# DYNAMIC CHARACTERIZATION OF A TALL BUILDING IN MONTREAL: INSIGHTS FROM AMBIENT VIBRATIONS AND NUMERICAL SIMULATION

Cyrielle Seymaux a*, Rola Assi, Ahmad Abo El Ezz, Guillaume Sieprawski

a* École de Technologie Supérieure,
1100 Rue Notre-Dame Ouest, Montréal, QC, H3C 1K3, Canada.
Courriel : Cyrielle.Seymaux@nck.ca
b École de Technologie Supérieure,
1100 Rue Notre-Dame Ouest, Montréal, QC, H3C 1K3, Canada.
Courriel : Ahmad.AboElEzz@etsmtl.ca
c École de Technologie Supérieure,
1100 Rue Notre-Dame Ouest, Montréal, QC, H3C 1K3, Canada.
Courriel : Rola.Assi@etsmtl.ca
d NCK Inc, Structural engineering consulting,
900-1200 Avenue McGill College, Montréal, QC, H3B 4G7, Canada.
Courriel : Guillaume.Sieprawski@nck.ca

Article soumis pour publication dans le journal *Canadian Journal of Civil Engineering*,
January 2026

## 3.1 Contexte de l'article

Cet article vient compléter le chapitre 2 en répondant aux deux dernières sous-problématiques de ce mémoire. Faisant suite à l'analyse modale in situ exposée précédemment, il propose une comparaison détaillée entre les résultats expérimentaux et ceux issus du modèle numérique de la tour, développé par l'entreprise NCK Inc.

Ce travail s'inscrit dans une démarche de variation paramétrique du modèle, calibré à partir de données mesurées sur une tour canadienne. Alors que peu d'études portent sur la calibration de modèles numériques à partir de mesures in situ pour des bâtiments de grande hauteur au Canada, cette contribution ajoute un cas local à une littérature majoritairement internationale. Les résultats apportent également aux firmes d'ingénierie des éléments concrets pour valider ou nuancer certaines hypothèses normatives, et mieux les intégrer dans leurs futures conceptions. Contrairement aux travaux se limitant à la calibration modale, cette étude pousse l'analyse plus loin en évaluant l'impact des paramètres calibrés sur les performances aux états limites de service (ÉLS), notamment en termes de déplacements inter-étages et d'accélérations. Une analyse de sensibilité est aussi menée, et une estimation de la section fissurée est proposée à travers la convergence des résultats expérimentaux et numériques.

Cet article soumis au *Canadian Journal of Civil Engineering* conclut le volet comparatif de l'étude de calibration, bien que ses résultats soient fondés sur une étude de cas, ce qui limite leur généralisation sans validation d'autres tours similaires.

## 3.2 Résumé

Cette étude porte sur la caractérisation dynamique et la calibration numérique d'un bâtiment de grande hauteur de 200 m récemment construit à Montréal. Les modèles aux éléments finis utilisés pour la conception reposent généralement sur des hypothèses conservatrices de rigidité et d'amortissement, susceptibles de diverger de la réponse mesurée in situ. Pour y remédier, des mesures d'ambiances vibratoires (MVA) ont été réalisées à l'aide d'accéléromètres disposés sur toute la hauteur du bâtiment. Les paramètres modaux expérimentaux, fréquences propres, taux d'amortissement et formes modales, ont été extraits à l'aide des méthodes de décomposition dans le domaine fréquentiel améliorée (DDFA) ou d'identification stochastique par sous-espaces (ISS). Des scripts Python ont ensuite permis de calibrer le modèle numérique en trois étapes : une analyse de sensibilité globale (ASG) des paramètres numériques, la projection des déformées modales à l'aide du procédé de réduction-expansion équivalente du système (SEREP), et la mise à jour du modèle. L'analyse de sensibilité a permis d'isoler les paramètres physiques les plus influents, ajustés pour réduire l'écart avec les résultats expérimentaux. Le modèle mis à jour reproduit les fréquences mesurées avec une précision de 5 % et atteint des valeurs de MAC supérieures à 0,9 pour les modes de flexion principaux, confirmant sa cohérence physique et numérique. Les amortissements mesurés in situ (0,8–1,2 %) sont nettement inférieurs à l'hypothèse de 2 % couramment utilisée en soufflerie, ce qui suggère une sous-estimation des accélérations et flèches de service. La calibration a également affiné les coefficients de fissuration : les murs du podium atteignent la limite physique d'environ 1,5, traduisant une demande résiduelle de rigidité liée au confinement des armatures. Cette étude enrichit la littérature canadienne sur la calibration in situ des modèles de tours et démontre le potentiel de ces approches pour améliorer la précision des analyses dynamiques et des prédictions en état de service.

Mots-clés : Bâtiment de grande hauteur, Vibrations ambiantes, Mise à jour de modèle, Modèle d'éléments finis, Analyse de sensibilité

## 3.3 Abstract

This study focuses on the dynamic characterization of a 200-meter high-rise case-study building recently constructed in Montreal. Finite element models of such buildings rely on conservative assumptions for stiffness and damping that can diverge from in-situ response. To address this, ambient vibration measurements (AVM) were conducted using accelerometers installed throughout the building height. Experimental modal parameters, including natural frequencies, damping ratios, and mode shapes, were extracted using Enhanced Frequency Domain Decomposition (EFDD) and Stochastic Subspace Identification (SSI). Python scripts were developed to calibrate a Finite Element Model (FEM) following a three-step procedure: a global sensitivity analysis to identify the most influential physical parameters, mode shape projection using the System Equivalent Reduction Expansion Process (SEREP), and a final FEM updating stage. Sensitivity analysis isolated the most influential physical parameters, which were adjusted to minimize discrepancies with measurements. Updated model reproduced the measured frequencies, confirming its consistency. In-situ damping ratios varied from 0.8 to 1.2% and were found to be lower than the 2% assumption typically used in standard design procedures. Stiffness modifiers that capture cracking-induced stiffness reduction for different types of structural elements are then proposed based on FEM updating. This study contributes to the limited literature on in-situ calibration and numerical modeling for tall buildings in Canada. It demonstrates the potential of such approaches to improve the accuracy of FEM and provides insights relevant to both international research and Canadian engineering practice.

Keywords: Tall building, Ambient vibrations, Model updating, Finite element models, Sensitivity analysis

## 3.4 Introduction

The dynamic characterization of high-rise buildings is increasingly important due to their sensitivity to wind-induced vibrations. During the design stage, finite element models (FEMs) of buildings are used to estimate structural response, especially for serviceability limit state (SLS) assessments, where accelerations and drifts under wind excitation are critical. These models often rely on normative and conservative assumptions for key parameters such as Young's modulus, moment of inertia, and structural damping, introducing uncertainty into a structure's actual dynamic properties. To improve the reliability of dynamic analysis in tall buildings, experimental modal testing is used to evaluate the in-situ dynamic properties of buildings that allow for more reliable identification of modal parameters such as mode shapes, natural frequencies and damping ratios, including torsional and coupling effects that are not necessarily well captured numerically with FEM. For in-situ assessment of dynamic properties of buildings under operational conditions, ambient vibration measurement (AVM) is a non-destructive technique that is widely used. Unlike forced vibration methods, AVM leverages natural excitations such as wind, traffic, and mechanical systems, making it particularly suitable for high-rise buildings. Modal parameters are extracted using signal processing techniques and are essential for FEM calibration.

Several studies have investigated and addressed the specific challenges of AVM in tall buildings, especially regarding signal synchronization and resolution. For example, Zhang et al. (2017) investigated the 632m Shanghai Tower using TROMINO® (Moho Science and Technology 2025) velocimeters placed along shear walls. Due to unreliable GPS signals indoors, sensors were synchronized externally before deployment. A 50-minute, 512Hz acquisition enabled the identification of eight low-frequency modes (~0.1Hz), with stable frequency estimates but significant uncertainty in damping ratios. Yun et al. (2020) demonstrated the critical role of frequency resolution in improved identification of modal and damping parameters in a 55-story building. Using a 500Hz sampling rate, they found that resolutions between $10^{-2}$ and $10^{-3}$Hz reliably captured all six vibration modes. However, a finer resolution of $2.5\times10^{-4}$Hz produced biased results for higher modes, as low-amplitude peaks

became harder to distinguish due to the increased number of spectral lines. To prevent such issues, Ventura et al. (2015) recommend selecting a sampling frequency at least 2.4 times the highest frequency of interest, ensuring adequate spectral peak separation and minimizing aliasing. Turek et al. (2007) conducted the modal analysis of a 44-story reinforced concrete building in Vancouver using a two-step decimation approach based on a 30-minute acquisition period: first to 100Hz for data processing, then to 5Hz for modal identification using the Enhanced Frequency Domain Decomposition (EFDD) and Stochastic Subspace Identification (SSI) approaches. Ten modes were identified below 5Hz, with consistent results across the methods, and damping ratios remained between 1% and 2%. Among these methods, SSI proved to be more accurate, particularly for damping ratios estimation. Yun et al. (2020) emphasized the importance of selecting appropriate measurement duration and frequency resolution when analyzing high-rise buildings. Their study shows that using time histories equivalent to 2000 times the fundamental structural period improves convergence of damping ratio estimates, especially in tall and flexible structures. They recommended a conservative minimum measurement duration of 3000 seconds, based on tests performed on 29-, 35-, and 55-story buildings, noting that shorter durations reduce modal peak amplitudes and may hinder the identification of higher modes.

A particularly important concern in the design stage of tall buildings is the assumption of a fixed 2% damping ratio. According to Canadian design standards (CNBC 2020, CSA A23.3-19), wind tunnel testing is recommended for high-rise buildings. These tests are commonly performed using rigid, reduced-scale models, and assume a fixed damping ratio of 2%. However, in-situ measurements on tall buildings often reported values below 1%, as noted in the CNBC Commentary I (CNBC - Division 4 2020). Smith and Willford (2008) studied many tall buildings with damping ratios below 1%. This overestimation of damping ratio can lead to underestimation of the dynamic responses such as floor accelerations and drifts, affecting the verification of serviceability criteria like occupant comfort and cladding performance. Similarly, stiffness assumptions based on reduced gross-section properties (typically 35% - 75%) may not necessarily capture the actual cracked-state behavior of reinforced concrete members, affecting the accuracy of predicted mode shapes and damping ratios (Brincker and

Ventura 2015). International standards such as Eurocode 8 (2004) and ASCE 7-22 (2022) adopt similar simplifications, typically assuming a 2% damping ratio. However, engineers commonly assume damping ratios between 1 % and 2 % for serviceability conditions, including wind-induced accelerations and drifts, with the ASCE 7-22 Commentary indicating that values within this range are typically adopted in the United States. Damping values close to 1 % are often selected because they reflect levels measured under ambient excitation and provide a conservative basis for serviceability checks.

In the context of dynamic characterisation of tall buildings, a global sensitivity analysis is typically conducted to evaluate how variations in physical properties influence these responses following the identification of modal parameters (natural frequencies, damping ratios, and mode shapes). Sensitivity analysis provides a structured framework to identify and rank physical parameters that most influence a structure's dynamic response by quantifying how input variability affects outputs such as natural frequencies and mode shapes. Global sensitivity analyses, such as the Morris or Sobol methods, are preferred over local methods such as local point-based methods (or gradient methods), for their ability to capture interactions across full parameter ranges (Razavi and Gupta 2015, Pan, et al. 2020, Brincker and Ventura 2015). The Morris method was selected in this study for its balance of efficiency and robustness (Razavi and Gupta 2015). It evaluates parameter influence by computing "elementary effects" over multiple trajectories, requiring relatively few model evaluations and computation time (Hurtado, Ortiz, et al. 2023, Tian 2018). An illustrative application of the Morris method is found in Pan et al. (2020), who updated the FEM of the Shanghai building using AVM data. The structural system of the building consists of a central reinforced concrete shear wall core as the primary lateral resistance. Outrigger and belt trusses connect this core to perimeter super columns, allowing lateral loads and overturning moments to be redistributed more effectively. The building was divided into nine structural zones corresponding to different occupancy types numbered from bottom to top (e.g., commercial, residential, mall, etc.). This zoning allowed for a more accurate and detailed assessment of the influence of physical parameters on the dynamic response along the height of the building.

The sensitivity analysis, based on finite differences (Pan and al. 2020), guided the selection of parameters by excluding those with low influence on the modal response. Results showed beams' density and Young's modulus had the most significant impact, while flexural inertias affected translational modes had minimal influence on torsional modes. Based on analysis of a 30-story coupled shear wall building, relatively flexible and structurally similar to both the Shanghai Tower and the tower examined in this study, Li and al. (2011) found that glazing contributed only marginally (~1.2 %) to the overall lateral stiffness. As a result, it was neglected in the present analysis.

FEM updating is then used to calibrate the physical parameters of the model, primarily Young's modulus (E), density (ρ), cracked section coefficients and moments of inertia (Ix, Iy), using data obtained from in-situ measurements. This process improves the accuracy of the model and provides insight into the effects of simplifying design assumptions, while enhancing the predictive capacity of structural performance assessments. The FEM updating process involves iterative adjustment of physical parameters to minimize discrepancies between experimental and numerical dynamic responses. Two strategies are considered (Wu and Li 2004): direct modification of system matrices (stiffness, mass) or indirect modification of physical properties. The latter is adopted in this study since it preserves the physical meaning of parameters and is better suited for tall buildings (Pan, et al. 2020). The procedure begins with a baseline FEM to compute initial modal frequencies and mode shapes. Experimental and numerical modes are then matched, and the model updating is formulated as a constrained least-squares problem. Parameters are updated iteratively until convergence. The quality of calibration is evaluated using the relative error between numerical and experimental natural frequencies, as recommended by Svendsen et al. (2020). Additionally, the Modal Assurance Criterion (MAC) is used to qualitatively assess the consistency between numerical and experimental mode shapes. A representative application is provided by Pan et al (Pan, et al. 2020), who updated the FEM of the 632m Shanghai building using AVM data. Parameters considered included beams' density (ρ), Young's modulus ($E$), cross-sectional area ($A$), and inertias ($Ix, Iy, J$), with typical variation bounds of ±25% to ±30%. The process reduced the average frequency error from 27% to 4%, while MAC values exceeded 90% for the first five

modes. Beam parameter adjustments included a ~25% reduction in torsional inertia, an 8.5% decrease in Young's modulus, and a 17% decrease in flexural stiffness in both principal directions, along with 13 - 15% increases in mass density and cross-sectional area. These results confirmed that the initial model overestimated the structure's stiffness, underlining the importance of model calibration based on field measurements.

This study aims to contribute to improved modelling and estimation of dynamic parameters of tall buildings using in-situ AVM and advanced modal updating techniques. AVM and FEM simulations were conducted on a case study high-rise building located in Montreal, Canada. Experimental frequencies, damping ratios, and mode shapes were extracted using EFDD and SSI techniques. To reduce computational effort and focus the calibration process on the most influential variables, a global sensitivity analysis (GSA) was performed before FEM updating. The aim is to identify the mechanical and geometric parameters that most affect the numerical structural dynamic response. While few studies have focused on calibrating numerical models using in-situ measurements for high rise buildings in Canada, this study contributes to international literature in this domain. The results also provide engineering firms with a standardised procedure to validate certain normative assumptions and better integrate them into their future designs. A sensitivity analysis is conducted, and an estimate of the cracked section modification factors is proposed through the convergence of experimental and numerical results.

### 3.4.1 Description of the studied building

The case study building is a 59-story residential and commercial tower located in Montreal, Canada, with three additional underground parking levels anchored into the rock substratum. The building, sensitive to and designed for wind load, reaches approximately 200m in height and is primarily composed of reinforced concrete (RC) structural elements. It features an irregular shape, with a podium at the lower stories and a more uniform configuration at the upper stories. The façade consists of glass panels, which minimally contribute to the building's lateral stiffness compared to the internal RC shear walls. The overall shape and configuration

of the building, as well as a schematic representation of the floor plans for the different structural zones (podium and tower), are shown in Figure 3.1. The lateral load-resisting system consists primarily of reinforced concrete shear walls with two distinct cores to provide lateral stability. The primary core extends continuously from the foundation to the top of the tower. The secondary core is limited to the podium levels, running from the foundation to the 11th floor. The columns serve to transfer gravity loads to the foundations and contribute minimally to the lateral resistance. At certain levels of the tower, coupling beams are used as part of the shear-wall system. The foundation system comprises ground-supported slabs, isolated footings, and strip footings. The three underground parking levels are embedded into the rock with 610mm thick retaining walls surrounding these levels considered in the FEM, resulting in minimal movements at these levels. Consequently, these levels have a limited impact on the building's dynamic response.

### 3.4.2 Ambient vibration measurements

The experimental setup involved the use of five TROMINO® velocimeters (TROMINO 2025), strategically positioned to capture both translational and torsional responses of the building. Sensors were installed near the shear core to record translational motion and at the slab extremities to better capture torsional behavior, as shown in Figure 3.1b-d. Figure 3.1a shows the zones 1 to 4, which correspond to the residential portion of the tower defined approximately every 10 stories to enable a more detailed sensitivity analysis along the building height. This zoning strategy supports more precise FEM calibration, as it allows for the selective inclusion or exclusion of parameters based on their influence. In this study, parameters showing less than 2.5% impact on the modal response are not considered in further FEM model updating. The podium zone (Zone 5) at the base of the structure is designated for commercial use.

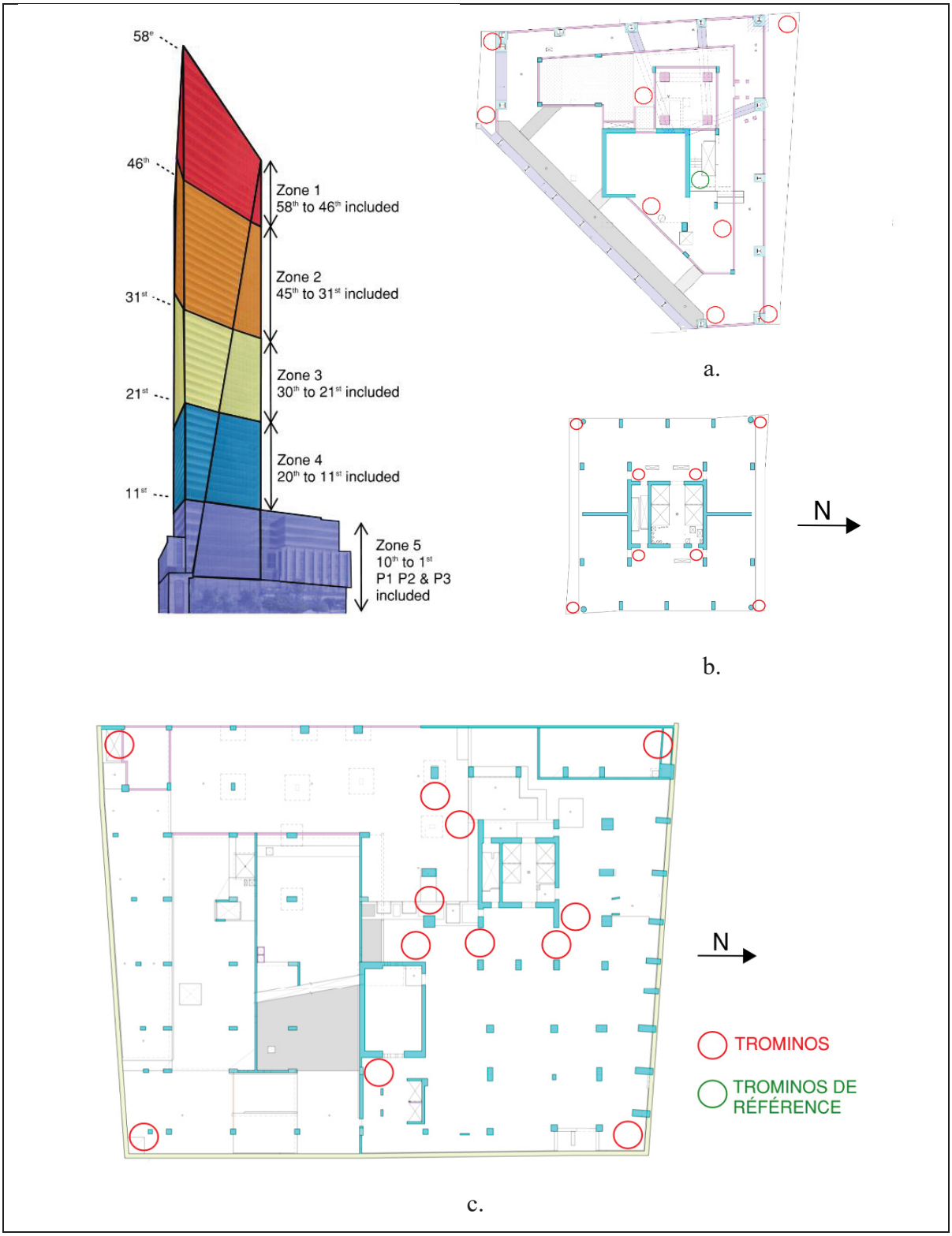



Figure 3.1 Sensors' localization along building's height: a) Top, b) Zone 2 to 4 and c) Zone 5

All devices were synchronized via radio since GPS signals could not penetrate multiple concrete slabs and shear walls. The recorded signals were stored and transmitted from the sensors to a computer using the GRILLA software (TROMINO 2025). This software allows to synchronize all sensor signals recorded via radio and ensures that each trace begins and ends at the same sensor internal clock time. To ensure accurate synchronization, all velocimeters were initially placed adjacent to the reference device on the 58$^{th}$ floor (the highest accessible level). Once synchronization was complete, the devices were relocated to their designated measurement floors. This repositioning resulted in a small initial time offset ($\Delta T$), which was removed using a custom Python script applied to the data exported from GRILLA. An example of the 35$^{th}$ story record is shown in Figure 3.2. Figure 3.2a still contains the time offset, whereas Figure 3.2b begins near the 500$^{th}$ second, after the offset has been removed. The resulting traces were then reformatted to be compatible with the ARTeMIS software (SVIBS 2025) for modal analysis.

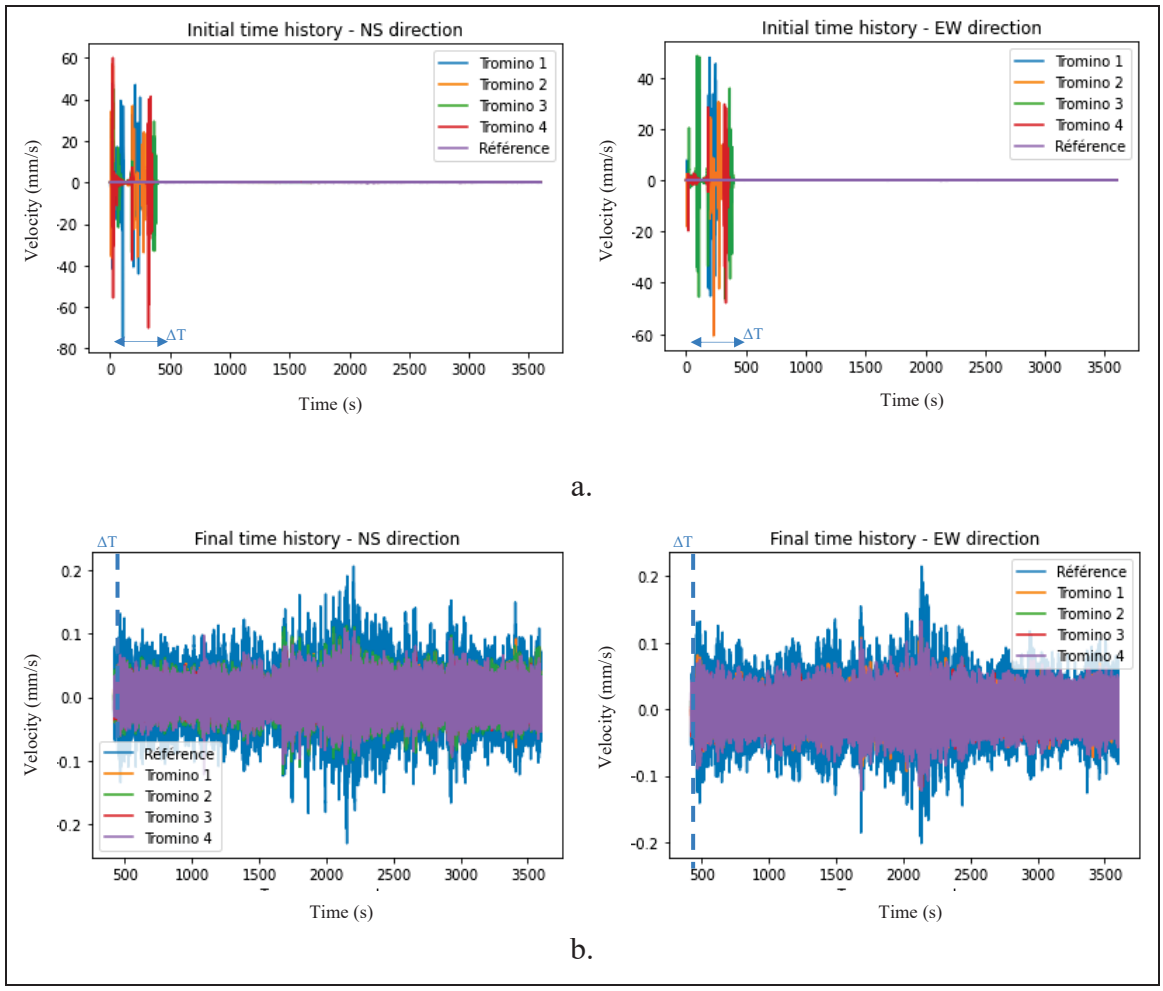



Figure 3.2 Initial signal example of the 35th story: a) Transl. 1 with Δt et b) without Δt in the NS and EO direction

The estimated fundamental frequency for the building is $f_1$ = 0.23 Hz and the damping ratio is $\zeta$ = 0.77%. Ventura (2015) recommends that, for high-rise buildings, the acquisition time should preferably satisfy the criterion T = $200/\zeta f_1$., where $\zeta$ is the modal damping ratio and $f_1$ is the fundamental frequency. However, for tall buildings with damping ratio below 1% and low fundamental frequencies (e.g., 0.23 Hz this criterion would require an acquisition time of approximately 31 hours per test, which is often cost-prohibitive and impractical. To address this, Ventura (2015) proposed a minimum duration defined as $T$min = $10/\zeta f_1$, which ensures enough damped cycles for reliable modal parameter estimation. Thus, the recommended minimum duration is approximately 5647 seconds (1.57 hours per test).

While this configuration aligns with recommended criteria in the literature for identifying low-frequency modes, it may be suboptimal for accurately estimating damping ratio values using frequency-domain methods such as EFDD. These methods typically require acquisition durations on the order of 2000 to 3000 times the fundamental period, depending on the building elevation and structural type (approximately 2.42 hours per test in this case). Nevertheless, due to logistical constraints, the acquisition time used in this study is 1 hour. These constraints include the need to minimize the duration of presence in individual residential units to avoid disturbing occupants, as well as completing the overall experimental campaign within a few months. Therefore, the data acquisition was conducted with a sampling frequency of 512Hz over a total duration of 1 hour (3600 seconds), yielding a frequency resolution of approximately $2.78\times10^{-4}$Hz, calculated as the inverse of the total acquisition time in seconds. This resolution is considered sufficient to reliably capture the first five modes required for numerical model calibration and damping ratio estimation. Signals were initially recorded at a sampling frequency of 512Hz. A first decimation to 256Hz was applied to reduce data volume while preserving the relevant frequency content. A second decimation to 2.54Hz was then performed, to focus the analysis on the frequency band of interest, up to approximately 1.27Hz (Nyquist frequency).

Although the final sampling frequency is slightly below the recommended threshold of $2.4 \times f_{\max}$ (i.e., 2.4 × 1 Hz = 2.4 Hz), the achieved frequency resolution is sufficiently fine for the accurate identification of the first five vibration modes and associated damping ratios, which are the most critical for numerical model calibration, especially since the fifth mode remains below 1Hz. This approach balances fidelity and efficiency, ensuring adequate capture of low-frequency and damping characteristics, while optimizing data volume and computational cost. The modal analysis was performed using ARTeMIS, applying both the SSI and EFDD methods. SSI methods are commonly used for output-only modal analysis and come in two main variants: SSI-COV (Covariance-driven) and SSI-DATA (Data-driven). SSI-COV is based on the covariance functions of the measured output signals and constructs a Hankel matrix from these correlations. It is more robust to noise and particularly well suited for ambient vibration data where the input excitation is unknown. In contrast, SSI-DATA directly

uses the raw time-domain response data to identify the system matrices, which can be more efficient computationally but is generally more sensitive to measurement noise. While results from both methods are presented for comparison, SSI, being a time-domain method, was selected for model updating, as it is more robust under short acquisition durations and less sensitive to spectral leakage and windowing effects, which can adversely affect frequency-domain approaches like EFDD.

### 3.4.3 FEM of the case study building

FEM of the studied building was developed in the ETABS software (Computers & Structures, Inc. 2025) as shown in Figure 3.3, which was verified against structural blueprints to ensure accurate geometry, stiffness, and mass distribution. Architectural DWG files were imported to refine element positioning, and 3D manual checks were conducted to resolve connectivity issues. The mass source included self-weight and an additional 70 kg/m cladding load, applied via fictitious frames with null mass and weight. To prevent numerical instabilities, these fictitious elements were assigned near-zero mechanical properties by applying stiffness modifiers with a factor of 0.1. Both lateral and vertical mass contributions, including rotational inertia, were considered. While vertical accelerations are not the primary focus, their inclusion allows the model to capture torsion-induced displacements, particularly at slab extremities. Additionally, the “Lump Lateral Mass at Story Levels” option was enabled to aggregate lateral mass between floor levels, ensuring a consistent distribution of modal masses.

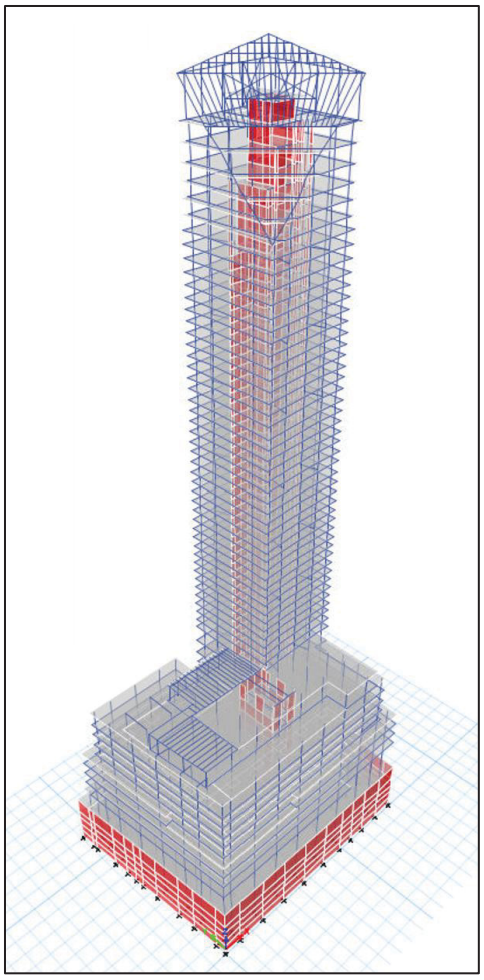


Figure 3.3 3D FEM of the studied building developed in ETABS®

Before any dynamic analysis, the FEM was validated using ETABS' check model by detecting disconnected joints, overlapping elements, mesh inconsistencies, and duplicated self-weight definitions. A modeling tolerance of 1 mm was enforced to ensure proper connectivity. After resolving these issues, multithreaded option analysis was enabled to accelerate computation times during the calibration process. Concrete elements were assigned material properties, cross-sections, and stiffness reduction factors according to CSA A23.3, to account for cracking effects (Tableau 3.1). The stiffness of columns, walls, slabs, and beams were adjusted accordingly. Moreover, the torsional constant $J$ of beams was set to 0.1 to prevent potential numerical instabilities related to beam torsional effects. Structural elements located in the basement levels were treated differently, as they are generally less affected by cracking due to

soil confinement, higher reinforcement ratios, and the nearly symmetrical lateral earth pressures acting on their faces. Unlike the superstructure, this uniform confinement limits crack development, an assumption that enhances numerical stability while preserving a realistic representation of the global structural stiffness.

Tableau 3.1 Stiffness reduction factors for serviceability
Taken from CSA Group A23.3-14 (2014)

| Member Type | Serviceability |
|---|---|
| Non-diagonally Reinforced Coupling Beams | 0.50 Ig / 0.40 Ag |
| Diagonally Reinforced Coupling Beams | 0.45 Ig / 0.45 Ag |
| Shear Walls | 0.95 Ig / 0.95 Ag |
| Shear Walls in net tension | Refined Calculation Required |
| Slabs with mild reinforcement | 0.35 Ig |
| PT slabs | 0.60 Ig |
| Beams (excluding coupling beams) | 0.50 Ig / 0.75 Ag |
| Columns | 1.0 Ig / 1.0 Ag |
| Columns in net tension | Refined calculation required for evaluating the coefficients |

Rigid diaphragms were initially used to determine the center of mass (CM) coordinates and to extract the modal masses at each story, which are essential inputs for the model calibration. After this step, the diaphragm configuration was changed to semi-rigid to more accurately represent the in-plane stiffness of the floor slabs, following the recommendations of Computers and Structures, Inc. (CSI) Wiki, the developer of ETABS® (Computers & Structures America, Inc. 2014). Then, manual nodes were created at the CM to facilitate displacement tracking. This modeling choice is considered a good engineering practice, as highlighted in CSI's guidelines, particularly when floor flexibility may influence the distribution of lateral loads

and the dynamic behavior of the structure. Consequently, all modal parameters, including natural frequencies and mode shapes, were extracted using the semi-rigid diaphragm model. In addition, boundary conditions were defined as pinned at the wall base, based on the assumption that the three basement levels are embedded in bedrock at the foundation level.

For dynamic characterization, Ritz vector analysis was preferred over Eigenvector analysis for its faster computational efficiency, as documented by Computers & Structures America, Inc. (2014). The extracted modes accounted for at least 90% of the total participating mass in each principal direction, thereby capturing the dominant dynamic behavior without unnecessarily increasing the number of modes. . The iterative analysis incorporated the mass P-Delta effect to account for geometric nonlinearity. The meshing strategy prioritized computational efficiency, applying a regular rectangular mesh instead of a general one with slightly coarser elements (3 meters by 3) in geometrically uniform areas to reduce analysis time while maintaining sufficient accuracy for modal calibration. While the model aims to match the actual tower as closely as possible, it remains a simplification, and slight variations in element dimensions or positions may exist. With all these assumptions validated and preliminary checks completed, the FEM is considered both structurally and numerically stable and is therefore suitable for subsequent modal analysis and FEM updating using in-situ data.

### 3.4.4 FEM updating methodology

The FEM updating process aims to reduce discrepancies between the numerical predictions and the observed dynamic behavior by calibrating key physical properties of the FEM using in-situ modal parameters. To ensure a meaningful comparison between experimental and numerical mode shapes, the SEREP (System Equivalent Reduction Expansion Process) method was employed and implemented using the PyFBS Python module. This approach enables the projection of measured in-situ modal displacements to equivalent degrees of freedom in the FEM, specifically at CM of each floor, thus allowing for a consistent and reliable calibration process. The methodology adopted in this study consists of several key steps. First, numerical mode shapes were extracted from the degrees of freedom (DOFs) of

interest, corresponding to the sensor locations used in the in-situ measurements as well as the CM at each story based on a semi-rigid diaphragm configuration. Both numerical and experimental mode shapes were then normalized independently using the maximum displacement amplitude in the X and Y directions. Next, the SEREP method was applied to extrapolate the in-situ modal displacements to the CM of each floor. From this point onward, only the mode shapes evaluated at the CM were retained for comparison. Both the experimental and numerical modal vectors were mass-normalized using the associated modal masses at the CM. With modal displacements consistently expressed and normalized at the CM, the FEM updating process was carried out using these refined values as input.

To reduce computational effort and focus the FEM updating process on the most influential variables, a global sensitivity analysis (GSA) was performed first. The aim is to identify the mechanical and geometric parameters that most affect the numerical structural dynamic response. Due to the complexity of numerical modeling, the contribution of the glazing system was not included in the sensitivity analysis, as its influence on lateral stiffness has been shown to be negligible in prior studies (Li, Hutchinson and Duffield 2011). The sensitivity analysis was conducted zone by zone, based on the structural segmentation shown in Figure 3.1a). For each zone, uncertain parameters such as Young's modulus $E$, moments of inertia $I_x$ and $I_y$ were varied within realistic bounds (Brincker and Ventura 2015, Pan, et al. 2020). Additional parameters, such as stiffness modifiers assigned to walls and slabs were also considered. These include membrane and bending stiffness modifiers in the local axes (e.g., $f_{11}$, $f_{22}$, $f_{12}$, $m_{11}$, $m_{22}$, $m_{12}$). They govern how cracking redistributes both global and local stiffness in the structure. These modifiers are capped at 1.0. Since they act as reduction factors for cracked sections, the stiffness cannot exceed the uncracked (1.0) baseline. Each parameter varies within realistic bounds-based literature information. The complete list of parameters and their variation ranges is presented in Tableau 3.2 based on the recommendation from one publication Pan and al and another Brincker and Ventura (2020, 2015).

Figure 3.4 illustrates the local axis convention, internal force resultants, and associated stiffness components of shell elements used to model slabs (horizontal shells) and walls

(vertical shells). For both element types, the local axes 1 and 2 lie in the plane of the shell, while the local axis 3 is normal to the element mid-surface. Membrane forces $f_{11}$, $f_{22}$ and shear force $f_{12}$ act in the plane of the element, whereas bending moments $m_{11}$ and $m_{22}$ represent flexural behavior about the local axes 1 and 2, respectively. These internal actions are directly linked to the in-plane and out-of-plane stiffness contributions of the shell, governed by the corresponding membrane and bending stiffness terms. The sign convention shown on the positive faces of the elements is consistently adopted throughout the numerical model to ensure a coherent interpretation of stiffness distribution, force transmission, and flexural response of slabs and walls.

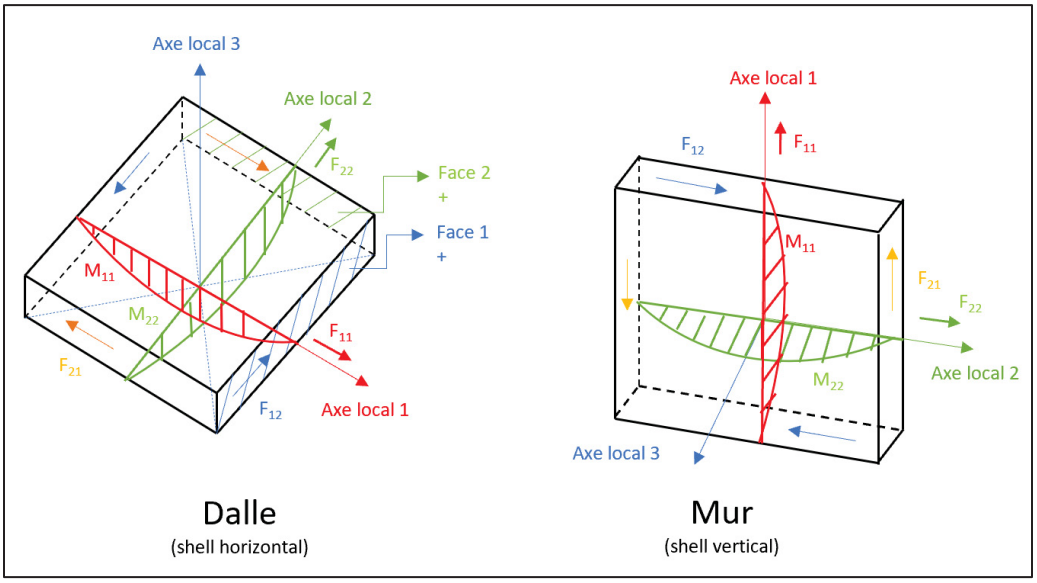



Figure 3.4 ETABS shell convention

Tableau 3.2 Defined ranges of mechanical properties for sensitivity and calibration analyses based on literature (**except for bolded values**)

Taken from Pan and al and Brincker and Ventura (2020, 2015)

| Parameter | Unit | Variation (%) | Zones concerned | Elements concerned |
|---|---|---|---|---|
| Young's modulus (E) (global stiffness) | GPa | –30 to +30 (limited to 46 GPa with rigid aggregates) | All zones | Columns, Beams, Slabs |
| Moment of inertia (Ix) | $m^4$ | –25 to +25 | All zones | Columns, Beams |
| Moment of inertia (Iy) | $m^4$ | –25 to +25 | All zones | Columns, Beams |
| In-plane stiffness modifiers ($f_{11}$, $f_{22}$, $f_{12}$) (local stiffness) | – | **–60 to +60 (limited to 1.50)** | All zones | Walls and slabs |
| Out-plane stiffness modifiers ($m_{11}$, $m_{22}$, $m_{12}$) (local stiffness) | – | **–60 to +60 (limited to 0.55)** | All zones | Slabs only |

The bolded values were refined during the calibration phase to ensure physically plausible parameter bounds consistent with the experimentally identified dynamic response, while also allowing an exploration of the upper stiffness local limits that the numerical model could realistically accommodate. In particular, the stiffness modifiers $f_{ii}$ and $m_{ii}$ were capped at 1.50 and 0.55, respectively, to preserve physical consistency. This apparent stiffness increase may be associated with higher reinforcement ratios, although these coefficients are generally expected to remain close to unity under predominantly compressive stress states with limited cracking.

The Morris method was selected for its effective balance between computational efficiency and parameter screening capability. The analysis was implemented using the SALib Python library and was linked to ETABS via its Application Programming Interface (API), which provides a programmable interface to control the software externally. This interface enabled automated parameter variation, model execution, and extraction of modal and dynamic response quantities without manual intervention. The results of the sensitivity analysis were used to eliminate parameters with negligible influence on the dynamic response of interest. Parameters contributing less than 2.5 % to the response variation were excluded from the subsequent optimization process to improve convergence speed. This pre-screening step significantly reduced the dimensionality of the optimization problem, making the model updating process more robust and computationally efficient.

The model updating procedure employed in this study follows a sensitivity-based iterative framework inspired by Svendsen et al. (2020), adapted for the calibration of high-rise building models using AVM data. A Python script, derived from Svendsen's open-source code was modified to interface with the ETABS® API and automate the full calibration loop. This includes updating selected parameters, launching modal analyses, extracting results, and computing error metrics. At each iteration, the numerical modal results are compared to the experimental ones at the floor CM to ensure consistent spatial alignment. This code is available in CALIBRATION : SCRIPT PYTHON section in the Annexes. Modal correlation is assessed using the Modal Assurance Criterion (MAC), where $\varphi_a$ represents the analytic mode shape and $\varphi_e$ the experimental one, below:

$$MAC_{(\varphi_a,\varphi_e)} = {|(\{\varphi_a\}^T\{\varphi_e\})|^2}\Big/{(\{\varphi_a\}^T\{\varphi_a\}\{\varphi_e\}^T\{\varphi_e\})} \tag{3.1}$$

As identified as the suitable method by Pan et al. (2020), the optimization targets the minimization of an objective function $J(\Delta\theta)$ *, formulated as a weighted least-squares error between measured and numerical natural frequencies shown below:

$$J(\Delta\theta)^* = \sum_{j=1}^{q} [W_\varepsilon]_{i,j} \left( {}^{z_{m,j} - z_{i,j}}/_{z_{0,j}} \right)^2 \tag{3.2}$$

Where, $\Delta\theta$ denotes the parameter increment vector, $q$ the measured outputs, $W\varepsilon$ the symmetric weighting matrix, $z_m$ the measured output, $z_0$ the initial output value and $z_i$ the FEM output occurring at each iteration. The weighting matrix $W$ is constructed as the squared inverse of the diagonal matrix formed from the measured natural frequencies. Specifically, each diagonal entry of $W$ is defined as $1/f_i^2$, where $f_i$ is the measured frequency of the $i^{th}$ mode. This approach assigns greater weight to errors associated with lower-frequency modes, which typically govern the global dynamic behavior of high-rise buildings. The resulting overdetermined system is solved using the bounded least-squares algorithm *scipy.optimize.lsq_linear*, which allows bounds to be enforced, preserving physical feasibility at every step. After convergence, either when a predefined error threshold is reached or the maximum number of iterations is completed, the final set of calibrated parameters is extracted. To assess the accuracy of the calibrated model, MAC matrices are compared before and after updating, highlighting the improvement in modal correlation. Lastly, a comparison table showing the physical parameters before and after the model updating is presented.

## 3.5 Estimation of in-situ modal parameters

The modal analysis using the EFDD method is shown in Figure 3.5. Figure 3.5a shows the singular values of the spectral density matrices for all setups, where prominent peaks indicate potential modal frequencies. The blue bands show the locations of harmonics detected during the analysis. At least eight dominant peaks can be observed within the 0 to 2.5 Hz range, corresponding to the expected fundamental vibration modes of the building. Figure 3.5b shows the average kurtosis computed across all projection channels. Kurtosis is a statistical indicator that quantifies the impulsiveness and peakedness of a signal. In operational modal analysis, high kurtosis values are typically associated with physical structural modes, while harmonic components and broadband noise generally exhibit lower kurtosis. Harmonics identified using the ARTeMIS harmonic indicator (SVIBS 2025), appear at regular frequency intervals and are

associated with periodic excitation sources rather than the intrinsic dynamic properties of the structure. As they do not represent physical eigenmodes governed by the mass–stiffness distribution, their inclusion would bias the identification of modal parameters and must therefore be excluded from the final modal set. In this case, multiple harmonic candidates emerge near 0.4 Hz intervals, underscoring the importance of validating the physical origin of each peak prior to its inclusion in the final modal identification.

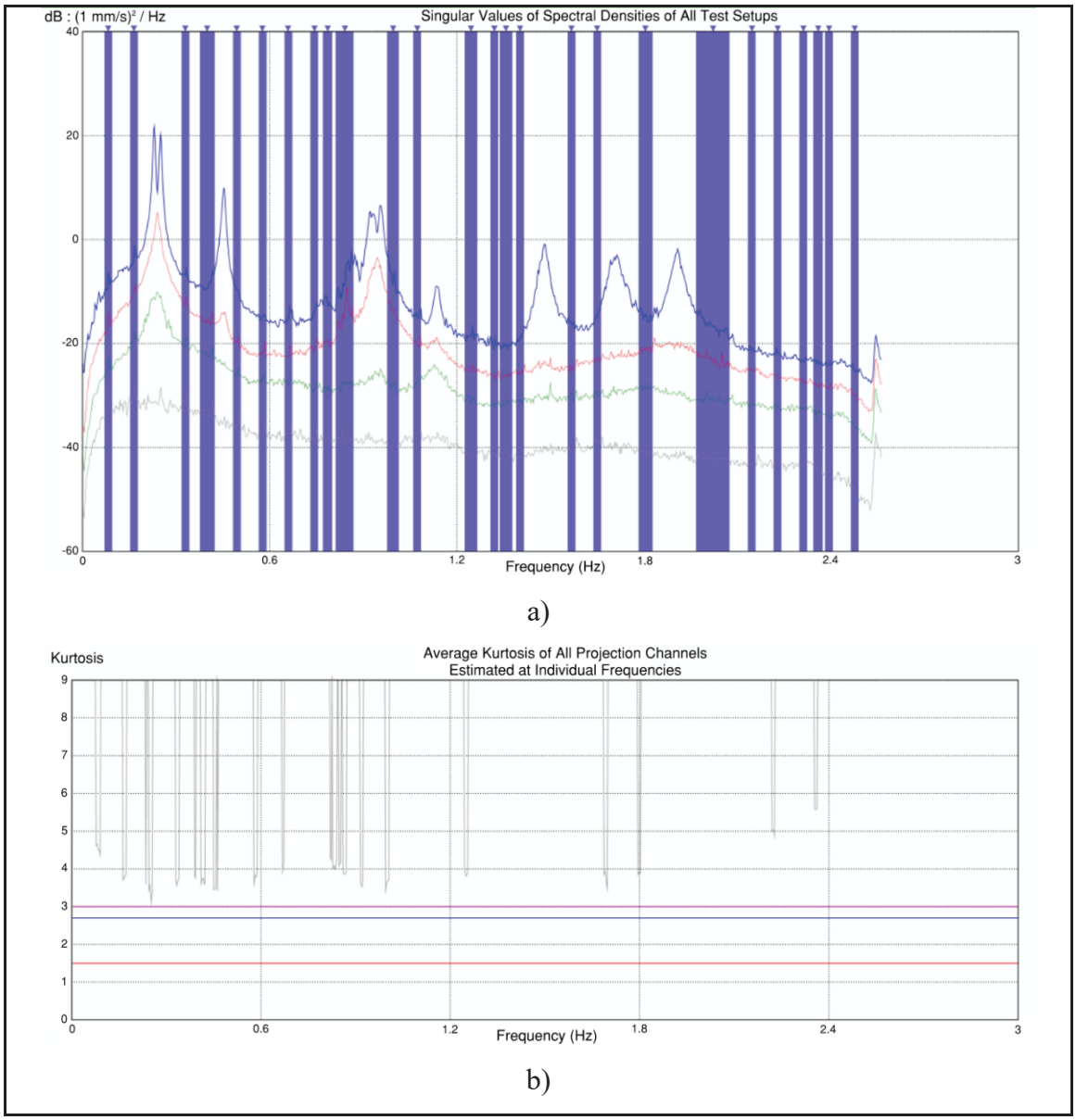



Figure 3.5 EFDD modal analysis results: a) Singular values of spectral density matrices and, b) Harmonic identification

To validate and complement the EFDD findings, a stabilization diagram based on the SSI method was generated for the same dataset, as shown in Figure 3.6. This diagram combines the singular values and a vertical alignment of stable poles across increasing model orders (y-axis). Stable poles are highlighted in red, and those that remain consistently aligned over a range of dimensions are interpreted as physical modes.

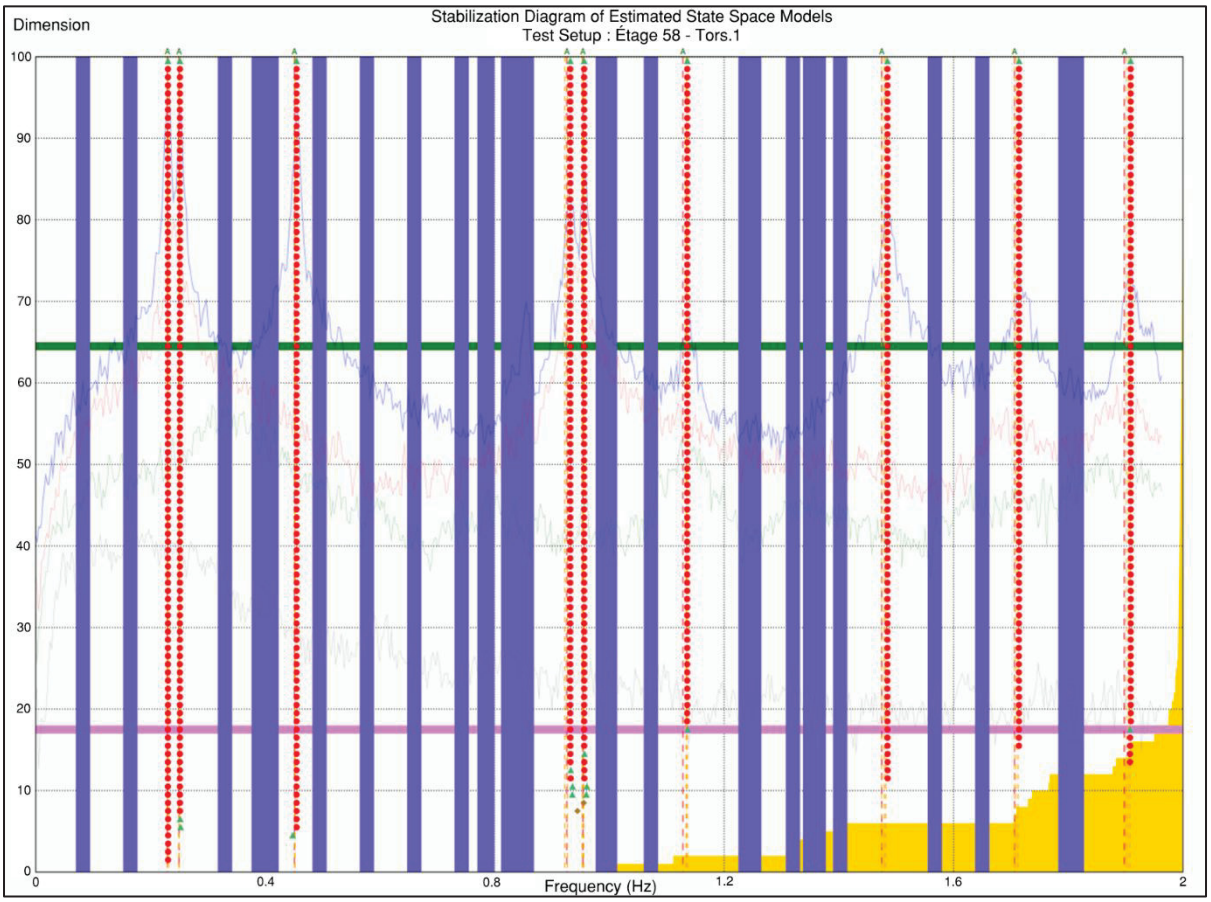



Figure 3.6 SSI stabilization diagram and harmonics

The harmonics previously identified in the EFDD spectrum are also observed in the SSI stabilization diagram, as indicated by the vertical blue bands. Unlike EFDD, the SSI method relies on a time-domain formulation and applies stability criteria on frequency, damping, and mode shapes, which allows physical modes to be distinguished more reliably from harmonics.

In contrast, the damping ratio estimates showed greater variability. EFDD systematically reported lower values, particularly for Modes 4, 5 and higher, due to its spectral formulation, which is more sensitive to noise and closely spaced peaks. This discrepancy is mainly attributed to the harmonics, which complicate the extraction of the resonance peak in EFDD's frequency-domain approach, as well as to the limited acquisition time, which may be insufficient to yield stable damping ratio estimates. In comparison, the SSI methods, which operate in the time domain, appear more robust against such disturbances. The combined use of EFDD and SSI thus enhances the reliability of the building's dynamic characterization. However, no significant difference in damping ratio stability can be confirmed between the SSI-PC and SSI-UPC variants, based on a single test campaign. To assess the accuracy of the calibrated model, MAC matrices of both methods. Figure 3.7 shows that both methods are accurate as 1 is on the diagonal and nearly 0 on the non-diagonal.

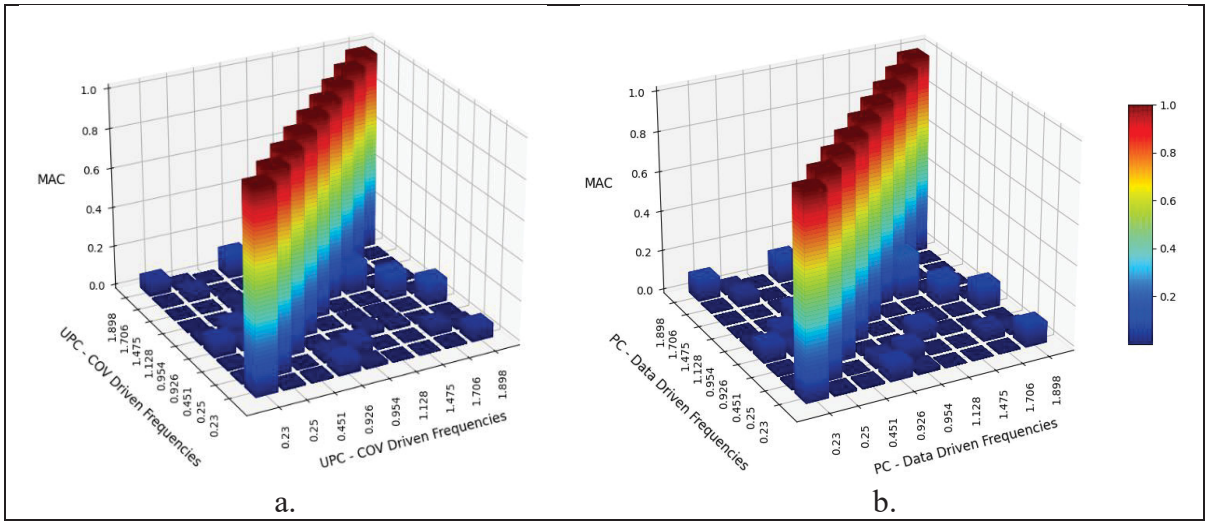



Figure 3.7 MAC matrix of SSI methods: a) UPC - COV Driven b) PC - Data Driven.

During ambient vibration testing, the dynamic response of tall buildings is excited mainly by wind. Consequently, the amplitude and directionality of the wind excitation can influence the parameters extracted through EFDD and SSI. Under stronger wind excitation, the building responds with larger vibration amplitudes, which tend to activate nonlinear mechanisms such as micro-cracking, for example. These nonlinearities cause an apparent reduction of stiffness, leading to slightly lower identified frequencies and higher equivalent damping ratios.

Conversely, under calm conditions, responses are smaller and mostly linear, yielding higher natural frequencies and lower damping estimates. Furthermore, wind excitation is broadband but directional. If the ambient wind is predominantly aligned with one façade, modes oriented in that direction are better excited, improving their observability, while transverse or torsional modes may appear weaker or noisier. This can affect the stability of identified damping ratios, especially for torsional or coupled modes that are less efficiently excited. This study could have been strengthened by concurrent anemometer data to quantify wind amplitude and direction during the tests. Such records would enable a direct comparison between the ambient wind conditions and the code service-level design wind and allow amplitude-normalized interpretations of the identified frequencies and damping.

## 3.6 Comparison with the initial FEM modal parameters

The modal identification conducted using EFDD, SSI-PC, and SSI-UPC yielded highly consistent frequency estimates across the first nine modes, despite the presence of harmonics as shown in Figure 3.8a. A comparison of the identified natural frequencies reveals a strong consistency between the three experimental methods (EFDD, SSI-UPC, and SSI-PC), with nearly identical values across all modes. This high level of agreement among the three methods confirms the robustness and reliability of the in-situ modal identification process. In contrast, the frequencies predicted by the initial ETABS numerical model are systematically lower than the experimental results. The discrepancies are particularly pronounced in the lower modes, with underestimations reaching up to 58% for mode 3. Even in higher modes, differences remain significant, typically between 20% and 40%. These differences suggest that the FEM overestimates the structure’s flexibility, likely due to conservative assumptions regarding material stiffness, cracked section properties based on the serviceability limit, and damping. These findings highlight the need for FEM updating to better capture the structure’s effective dynamic behavior. Overall, strong agreement between experimental methods reinforces confidence in the identified modal parameters, which provide a reliable reference for FEM updating.


Figure 3.8 Comparison of in-situ and initial numerical modal analysis results a) frequencies and, b) damping ratios

## 3.7 Results of the sensitivity analysis

The 3D MAC matrices provide clearer justification for method selection. As shown in Figure 3.7, the SSI-UPC method (Figure 3.7a) exhibits high modal consistency, with diagonal MAC values close to 1 similar to SSI-PC (Figure 3.7b), but achieves a better mode separation,

particularly evident for the 1.475Hz mode, where off diagonal values are closer to zero . This indicates more accurate and orthogonal mode shapes, which is particularly important for FEM updating that relies on well-identified and orthogonal modal vectors. Therefore, for the remainder of the study and the FEM updating process, only the first five modes identified using the SSI-UPC method are considered. The fifth mode has a frequency of 0.95Hz, which remains below 1 Hz and is well separated from the next mode at 1.40Hz. This selection satisfies decimation guideline proposed in Brincker and Ventura (2015). The decimated sampling frequency used in this study is 2.56Hz, which is slightly above the recommended threshold of $2.4 \times f_{max}$ (i.e., $2.4 \times 1$ Hz = 2.4Hz), thus justifying the selection of the first five modes. The resulting frequency resolution remains very fine, making it well suited for the accurate identification of both modal frequencies and damping ratios, which are key parameters for reliable numerical model calibration.

The figures below present the normalized sensitivity indices (μ*) for all studied parameters along the building height. The different colors used match the zones' separation described earlier. Each cell indicates the normalized mean effect (μ*) of a specific parameter on the identified modal frequencies, expressed as a percentage. Higher μ* values reflect greater influence of the corresponding parameter on the frequencies and mode shapes. The beams at the lower podium levels are neglected, since the podium is much stiffer than the more flexible building above. Likewise, the coupling beams within the building are limited in number, so their contribution to the global stiffness is considered negligible.

**Upper zone 1:**

In the upper zone of the building (floors 46 to 58 - red zone), shown in Figure 3.9, slab- and wall-related parameters dominate the dynamic response. The slab Young's modulus shows the strongest influence (μ* > 35% for Modes 1 and 2 and still > 20% at Mode 5). In contrast, the wall Young's modulus is minor for Modes 1 and 2 but rises to ≈30% for the higher modes. This pattern is consistent with larger modal amplitudes in the upper stories, where slab mass and in-plane stiffness govern the global bending response. As torsional contribution increases

with mode order, both in-plane ($f_{ii}$) and out-of-plane ($m_{ii}$) stiffness of walls and diaphragms are mobilized, explaining the growing wall contribution.

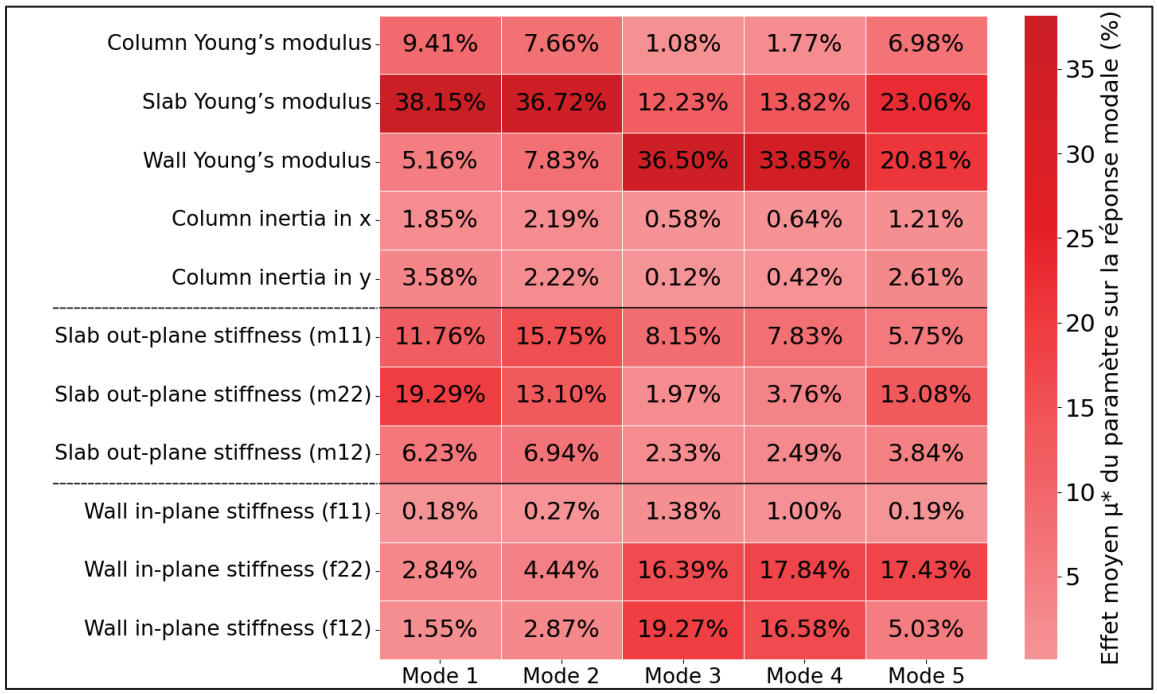



Figure 3.9 Average affect μ* of parameters on the modal response: Zone 1

**Intermediate zones 2 and 3:**

In the intermediate zones (Zones 2 and 3 - orange and yellow zones represented by Figure 3.10a and Figure 3.10b respectively), the response shifts progressively from diaphragm-influenced bending to wall-controlled behaviour. In Zone 2, slab $E$ remains prominent in Modes 1-2 (≈ 25%) but drops to 12% in Modes 3-5, while wall $E$ rises from 20% to 40%. Slab $m_{22}$ matters at Mode 1 (≈ 12%) then fades, whereas wall in-plane $f_{22}$ climbs from 8% to 41%. In Zone 3, the trend is accentuated. Wall $E$ is already high at low modes (≈ 30%) and dominates Modes 3-5 (≈ 45%). Wall $f_{22}$ similarly increases from 25% to 45%. By contrast, slab $E$ declines from ≈20% (Mode 1) to ≈1% (Mode 5). Finally, column inertia terms remain minor. Physically, increasing torsional content with mode order, engaging wall membrane ($f_{ii}$) and out-of-plane action ($m_{ii}$), so wall properties progressively control the stiffness while diaphragm effects become secondary.

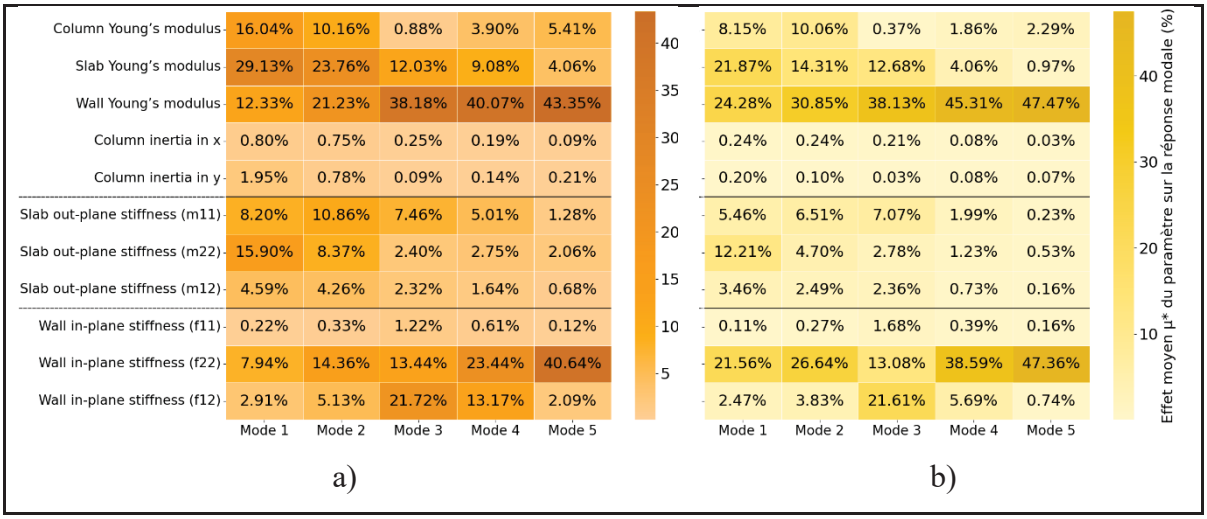



Figure 3.10 Average affect μ* of parameters on the modal response: a) Zone 2 and b) Zone 3

**Lower zones 4 and 5:**

In lower zones (Zones 4 and 5 - blue and purple zones represented by Figure 3.11a and Figure 3.11b respectively), wall $E$ governs across modes (≈ 40%), with wall in-plane $f_{22}$ of similar magnitude and a marked coupling $f_{12}$ at Mode 3 (≈22%). Slab $E$ stays secondary (≈10%), column $E$ is only notable in Mode 1 and 2 (≈16%), and slab $m_{ii}$ terms are low. In the podium (Zone 5), walls still dominate but columns contribute more to the lowest modes (≈25% in Mode 1 and 2). Slab $E$ remains small (≈8%), $m_{ii}$ modest (≈5%), and wall $f_{12}$ appears chiefly at Mode 3 and 4 (≈13%). Overall, diaphragms act mainly as collectors, while shear walls control global stiffness (membrane $f_{ii}$ and $m_{ii}$ out-of-plane), with column participation reinforced at the podium. In summary, the global sensitivity analysis highlights the dominant influence of slab and wall parameters on the dynamic behavior of the tower. Accordingly, although several parameters exceeded the screening threshold (μ* > 2.5%), the calibration was restricted to the Young's moduli $E$ of walls and slabs. The stiffness modifiers $m_{ii}$ (out-of-plane) and $f_{ii}$ (in-plane), which are intended to model cracking (i.e., stiffness reduction), were not calibrated. Indeed, during optimization they tend to drift toward or even exceed 1.0 to compensate global stiffness, which is non-physical for cracked sections. Therefore, $m_{ii}$ and $f_{ii}$ are fixed at their initial normative-based (Tableau 3.1) values and only $E$ is considered for updating, avoiding

double-counting of stiffness and non-physical compensation while targeting the most meaningful material property.

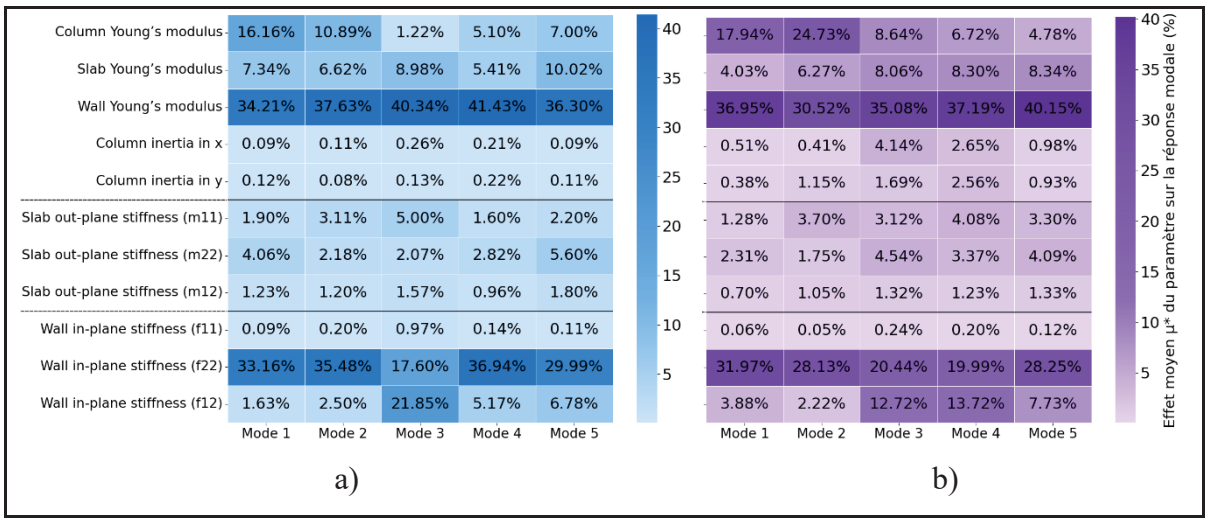



Figure 3.11 Average affect μ* of parameters on the modal response: a) Zone 4 and b) Zone 5

## 3.8 FEM updating results

Before proceeding with the FEM updating, Figure 3.12 compares the initial mode shapes from the FEM (evaluated at the center of mass of each floor) and the experimental mode shapes, also reconstructed at the center of mass using the SEREP method. This projection makes it possible to express the experimental measurements at the same spatial locations as the numerical results, enabling a direct and meaningful comparison. The first five modes in both the X and Y directions are considered, and all mode shapes are mass-normalized to eliminate the influence of absolute amplitude and focus solely on the shape profiles. A mass-normalized mode shape is defined such that its generalized modal mass is equal to unity, i.e. $\phi_i^T M \phi_i = 1$, where $\phi_i$ is the mode shape vector and $M$ is the mass matrix. Under this normalization, the modal amplitude is rendered dimensionless and no longer represents an absolute displacement level, but solely the relative spatial distribution of deformation. This normalization removes the arbitrary scaling inherent to numerical eigenvalue extraction and experimental identification, allowing consistent and meaningful comparisons between mode shapes. The first two modes (Figure 3.12a-b) show an overall agreement between the

experimental and numerical trends, particularly in the upper stories. This consistency suggests that the experimental mode shapes reconstructed at the center of mass are reliable and sufficiently accurate to serve as reference targets for the calibration process. Noticeable discrepancies are observed in the intermediate and lower floors, which is expected, since the initial model, although it includes code-based stiffness reduction factors, does not necessarily reflect the actual cracking state of the structure observed in-situ. Higher-order modes (Figure 3.12c-e) exhibit more pronounced differences, especially in terms of curvature and nodal locations. These modes may involve torsional components or flexural–torsional coupling effects, which are sensitive to local variations in stiffness and mass distribution. Despite these differences, the SEREP reconstruction provides spatially consistent experimental mode shapes, enabling a structured comparison with the FEM. In summary, this preliminary comparison does not aim to assess the accuracy of the initial FEM, but rather to validate the precision of the reconstructed experimental mode shapes. Their overall coherence at the center of mass confirms that they can be reliably used as targets in the upcoming model updating phase.

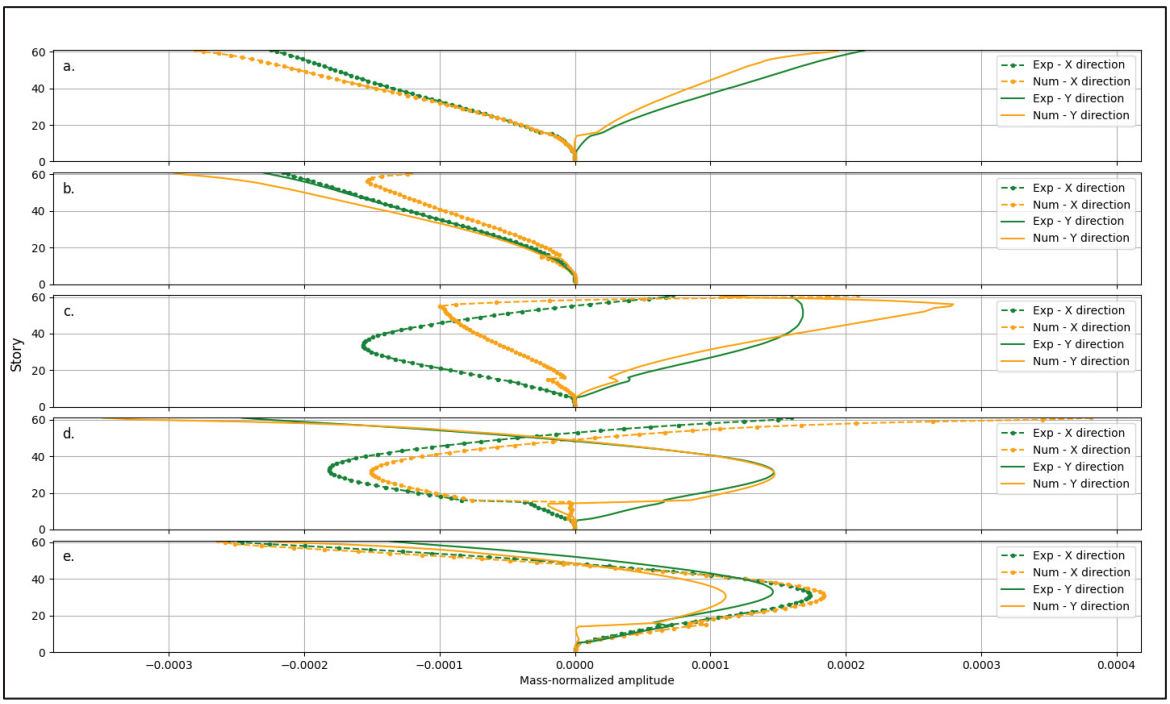



Figure 3.12 Initial numerical and experimental (SEREP) mode shapes at story mass centers: a) Mode 1 to e) Mode 5

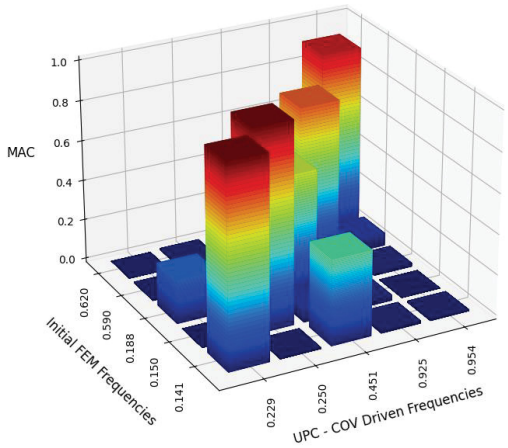
Tableau 3.3 below summarizes the measured, initial numerical, and updated frequencies for the first five modes, as well as the relative errors and Modal Assurance Criterion (MAC) values before and after calibration.

Tableau 3.3 Initial and updated frequencies and MAC values (Modes 1 to 5)

| | Frequency (Hz) | | | | | MAC | | |
|---|---|---|---|---|---|---|---|---|
| Mode | Measured | Initial | Initial Error (%) | Updated | Updated Error (%) | Initial MAC | Updated MAC | MAC Δ (%) |
| 1 | 0.229 | 0.141 | -38.43 | 0.217 | -5.29 | 0.997 | 0.999 | 0.222 |
| 2 | 0.250 | 0.150 | -40.00 | 0.242 | -3.23 | 0.998 | 0.999 | 0.050 |
| 3 | 0.451 | 0.188 | -58.31 | 0.453 | 0.51 | 0.607 | 0.579 | -4.706 |
| 4 | 0.925 | 0.590 | -36.22 | 0.933 | 0.84 | 0.798 | 0.860 | 7.816 |
| 5 | 0.954 | 0.620 | -35.01 | 0.965 | 1.12 | 0.916 | 0.929 | 1.424 |

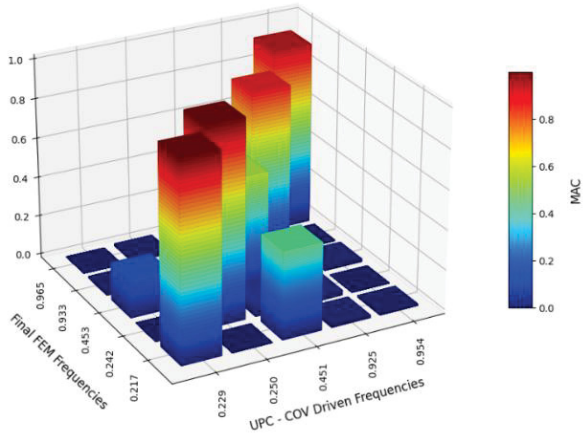
Tableau 3.3 compares the measured and numerical frequency and mode shapes precision (MAC) before and after FEM updating. The initial FEM underestimated all measured frequencies by approximately 35–60%. After calibration, the updated frequencies closely matched the measured ones, with residual errors below ±5%, confirming a successful global stiffness adjustment. The Modal Assurance Criterion (MAC) values also improved for most modes, remaining above 0.9 for the first two bending modes, which demonstrates good agreement in mode-shape patterns. Mode 4 shows a clear improvement from 0.798 to 0.860 (+7.8%), and Mode 5 also slightly increases. In contrast, Mode 3 shows a decrease in MAC (–4.7%), which is expected since this mode is torsional. Torsional modes are typically more difficult to capture accurately from AVM due to lower excitation energy and limited sensor sensitivity in rotational motion. Small phase differences or measurement noise can significantly affect their identified shape correlation, even when frequency matching is satisfactory. Overall, the updated model reproduces the measured dynamic behavior with high fidelity, particularly for the translational modes governing the building's global response.

To assess the accuracy of the calibrated model, MAC matrices are compared before and after FEM updating are shown in Figure 3.13, highlighting the improvement in modal correlation. Lastly, a comparison table showing the physical parameters before and after the FEM updating is presented. Figure 3.13a presents the initial MAC matrix computed between the uncalibrated FEM mode shapes and the SEREP-reconstructed experimental shapes, while Figure 3.13b shows the MAC after updating. Each 3D bar represents the MAC value between a pair of modes (experimental vs numerical), with color indicating correlation strength.




| Hz | 0.229 | 0.250 | 0.451 | 0.925 | 0.954 |
|---|---|---|---|---|---|
| **0.141** | 0.9966 | 0.0075 | 0.4252 | 0.0152 | 0.0091 |
| **0.150** | 0.0004 | 0.9982 | 0.0609 | 0.0275 | 0.0039 |
| **0.188** | 0.2159 | 0.0511 | 0.6073 | 0.1552 | 0.0194 |
| **0.590** | 0.006 | 0.0003 | 0.1861 | 0.7976 | 0.0656 |
| **0.620** | 0.0011 | 0.0102 | 0.0476 | 0.0234 | 0.9164 |

a)




| Hz | 0.229 | 0.250 | 0.451 | 0.925 | 0.954 |
|---|---|---|---|---|---|
| **0.217** | 0.9988 | 0.0034 | 0.4414 | 0.0157 | 0.0102 |
| **0.242** | 0.0019 | 0.9987 | 0.0547 | 0.0268 | 0.0038 |
| **0.453** | 0.1936 | 0.0494 | 0.5787 | 0.1459 | 0.0174 |
| **0.933** | 0.0096 | 0.0 | 0.2316 | 0.8599 | 0.0054 |
| **0.965** | 0.0003 | 0.0105 | 0.0134 | 0.0004 | 0.9295 |

b)

Figure 3.13 Modal assurance criterion between experimental mode shape and a) initial model (SEREP) and b) final updated model

The first two modes are very well identified in both cases. Before updating (Figure 3.10a), strong diagonal values (> 0.95) are observed only for the first two bending modes, indicating good agreement for the fundamental translational responses. However, off-diagonal coupling remains visible, particularly around the third and fourth modes, suggesting imperfect mode separation and limited correlation of higher modes. After updating, the MAC matrix becomes more diagonal, and the correlation of the identified mode shapes improves overall. The first two modes maintain near-perfect agreement (MAC ≈ 0.999), and the fourth and fifth modes also show a noticeable enhancement, reaching values above 0.85-0.92. Mode 3 (torsional) stays the least correlated (≈ 0.58), a known challenge in AVM due to lower torsional energy and higher sensitivity to measurement or modeling asymmetries. Nevertheless, the off-diagonal terms generally decrease, evidencing cleaner one-to-one pairing. Overall, the updated model better reproduces the measured dynamics, with improved modal alignment and reduced cross-correlation; the remaining torsional mismatch likely reflects minor stiffness asymmetries, simplified diaphragm modeling, or residual flexural–torsional coupling.

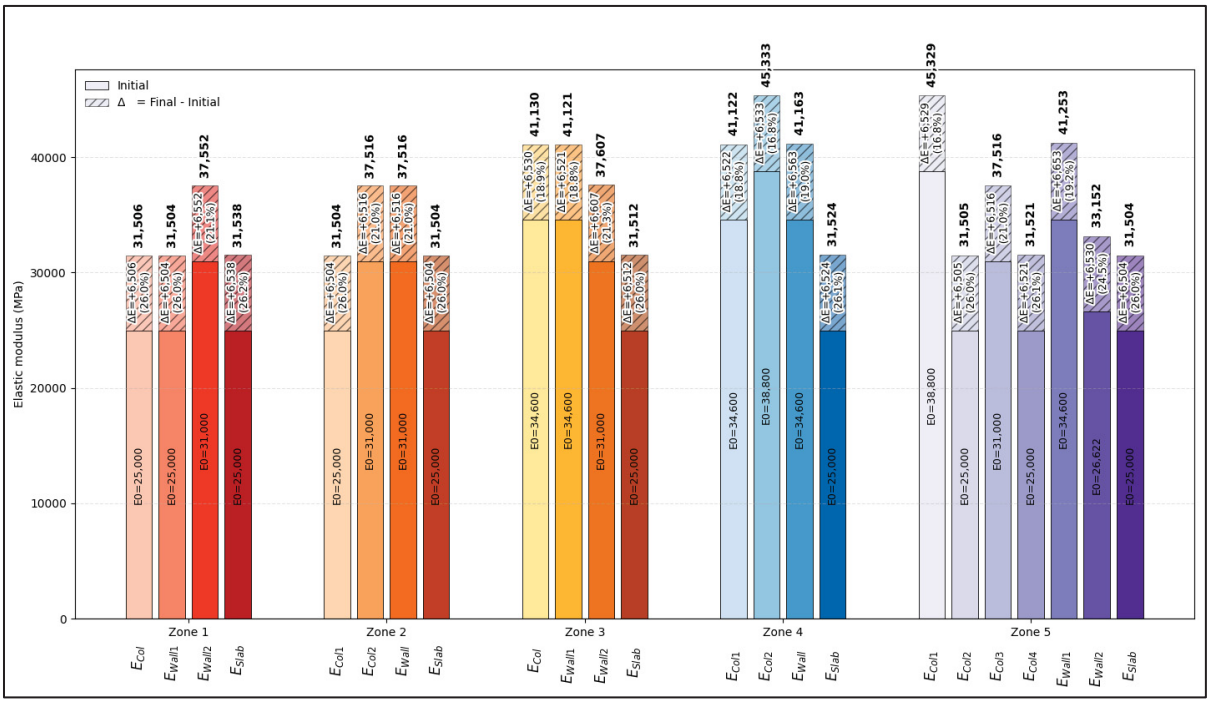



Figure 3.14 Young moduli variations by zone after FEM updating

Figure 3.14 presents the results of the global updating performed on the Young's modulus across the five structural zones. A clear vertical trend emerges. The calibrated moduli increase from the top (Zone 1) to the podium (Zone 5), reflecting the growing stiffness and confinement toward the base of the tower. The upper zones converge around 31-37 GPa, while the podium reaches values up to ~ 45 GPa, consistent with the higher concrete grades and reinforcement density expected in the foundation and transfer regions. As this updating was conducted globally trough E variations, it primarily corrected the overall stiffness deficit observed in the initial model. The resulting distribution of $E$ now better captures the physical behavior of the structure, with a realistic stiffness gradient along the height of the tower: more flexible at the top and significantly stiffer near the base. This confirms that the global updating effectively rebalanced the tower's stiffness profile while preserving its modal characteristics.

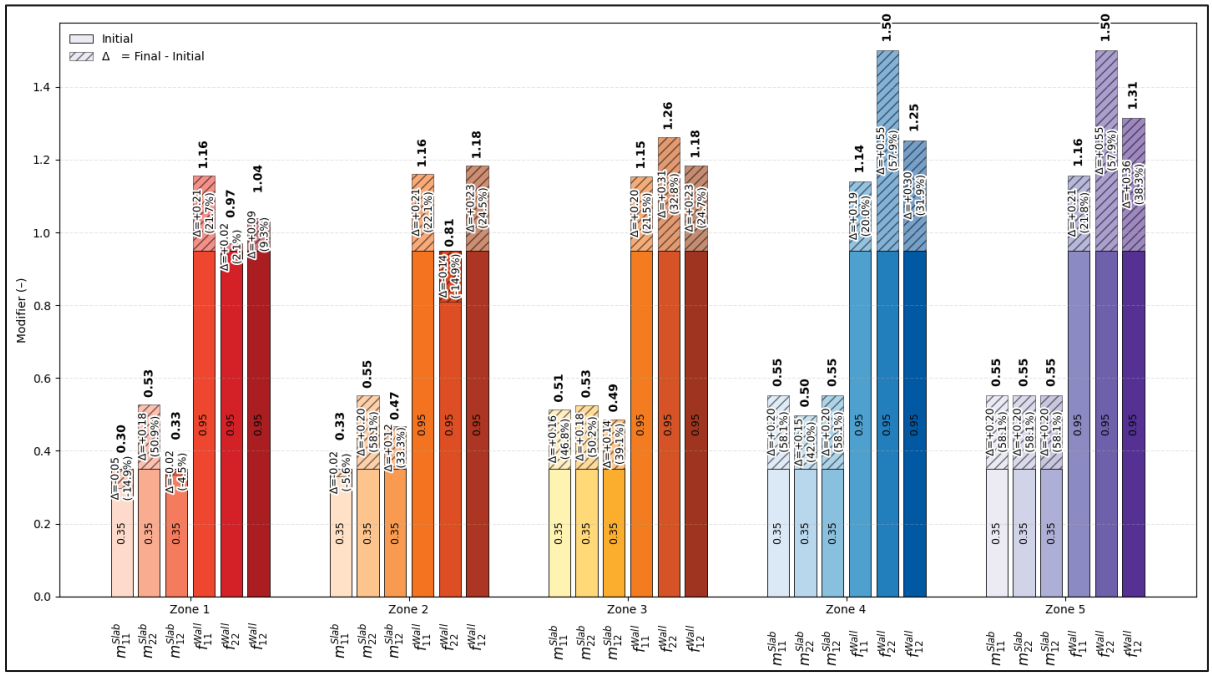



Figure 3.15 Local proprieties' modifiers variation by Zone after FEM Updating

Figure 3.15 shows the local updating of the slab and wall stiffness modifiers. The initial design model used conventional cracked-section coefficients of 0.35 for slabs and 0.95 for walls, as typically prescribed by design codes for serviceability conditions. After updating, the results show that the loss of stiffness is limited to the top of the tower (Zone 1) and affects mainly the

in-plane slab directions ($m_{ii}$), which remain below unity. For this zone, this outcome is physically consistent with code-based expectations: upper slabs experience greater tensile strains and flexural cracking under lateral sway, leading to a lower effective stiffness. In contrast, wall modifiers remain close to or slightly above 1 along most of the height, which is logical since shear walls are predominantly under compression and therefore exhibit limited cracking. At the base (Zone 4, tower and Zone 5, podium), modifiers reach the upper bound of 1.5, reflecting the combined effects of dense reinforcement, confinement, and transfer-level stiffening, rather than unrealistically high elastic behavior. Exceeding this threshold would not be physically meaningful, as modifiers represent cracked-section reductions relative to uncracked stiffness. This also shows that the numerical model tends to demand even more rigidity at its bottom as it wants to go higher than 1.5. Overall, this local updating confirms the physical relevance of the modifiers initially used by designers: the 0.35/0.95 assumptions remain accurate at the top, where cracking governs, while the calibrated increase toward the base captures the real stiffness gain from reinforcement congestion and limited cracking in the lower structural zones. The model thus achieves a realistic stiffness gradient, from flexible diaphragms aloft to stiffer, compression-dominated regions near the foundation. Finally, wind-tunnel loads are typically supplied for an assumed structural damping of 2 % and rigid system. In-situ measurements indicate lower damping (~ 0.8–1.2 %), so using the 2 % assumption underestimates the dynamic response at SLS. Consequently, accelerations and drifts, particularly at the top, are likely higher than predicted, which can affect occupant comfort and non-structural performance.

## 3.9 Conclusions and recommendations

This study contributes to the still limited body of literature on in-situ calibration of numerical models for tall buildings in Canada. It highlights the potential of this approach to enhance the fidelity of numerical predictions and offers practical insights for both international research and the Canadian engineering community. In this work, the dynamic characterization and FEM updating of a tall concrete building located in Montreal were carried out using in-situ AVM. Modal parameters were extracted using both frequency-domain (EFDD) and time-domain

(SSI-COV) techniques and compared to the response of a initial FEM developed in ETABS using code-based stiffness reduction factors for structural elements. Through sensitivity analysis, the FEM was iteratively updated by adjusting key physical and geometric parameters for each structural zone. The final updated FEM showed good agreement with measured frequencies, with relative errors near 5%, and good mode shape correlation, especially for modes 1, 2 and 5 (MAC > 0.9). In contrast, modes 3, which involve torsional and coupled lateral-longitudinal behavior, exhibited lower MAC values, reflecting the greater challenge of experimentally identifying these deformation patterns due to sensor distribution limitations and reduced torsional energy in ambient excitation.

The FEM updating revealed a consistent stiffness along the height of the tower, with Young's moduli increasing from 31–37GPa in upper levels to ~45 GPa in the podium. These values align with material confinement and reinforcement density observed in practice. Locally, the stiffness modifiers of slabs and walls converged toward physically meaningful ranges, like ~0.35 for cracked slabs and ~1.0 for walls in compression, validating the normative CSA A23.3 assumptions used in design. However, the need for higher stiffness (up to 1.5) near the base suggests additional rigidity due to reinforcement congestion or transfer-level effects not fully captured in design models. These findings emphasize the need to reconsider code-based assumptions when accurate service-level performance is required, particularly in tall and flexible buildings. A particularly important observation concerns the assumption of a fixed 2% damping ratio for wind design, as commonly used in wind tunnel studies and design codes. The experimental results reveal that actual damping ratio is frequently below 1%, especially for the fundamental modes. For wind-governed buildings, this overestimation of damping ratio can impact the predictions of peak accelerations for serviceability assessment.

This study demonstrates the value of combining experimental modal analysis and FEM updating for real buildings, providing not only validation of design assumptions but also data-driven refinements for future simulations.. To improve the reliability of dynamic analyses in tall buildings, particularly those governed by wind loads, several recommendations emerge from this study. Experimental modal testing should be systematically integrated into design

validation after construction, allowing the development of in-situ databases. This is especially relevant for serviceability limit state (SLS) assessments, where wind-induced accelerations and drifts are governing criteria. These tests allow for a more accurate identification of modal parameters, including torsional and coupling effects that are often poorly captured numerically. However, the precision of these experimental results depends on appropriate data acquisition strategies. Extended acquisition times improve damping estimates using EFDD, while time-domain approaches such as SSI yield more robust results overall. A high frequency resolution must also be maintained, and signal decimation should remain above 2.54Hz to capture higher-order modes (e.g., 6th and above), which are essential for wind-sensitive structures. While limited to one case study, this methodology could be replicated across other Canadian buildings. To address this, future efforts should focus on refining experimental protocols generating a regional database of in-situ stiffness values and damping ratios to support better-informed design practices.

# CONCLUSIONS

Ce travail illustre, à travers l'étude d'un cas réel situé au centre-ville de Montréal, une caractérisation modale *in situ* par vibrations ambiantes (DDFA/ISS), comparée à un MÉF développé sous ETABS, puis une calibration progressive vers les résultats *in situ* de ce dernier guidée par une analyse de sensibilité. La chaîne méthodologique complète, depuis les mesures *in situ*, l'extraction des paramètres modaux par les approches DDFA/ISS, la comparaison des fréquences et des déformées modales, jusqu'à la mise à jour incrémentale du modèle numérique, a permis d'aligner de manière cohérente la réponse dynamique du modèle sur les observations expérimentales, consolidant ainsi une base numérique fiable pour les vérifications aux états limites de service. L'analyse modale *in situ* a notamment permis de quantifier avec précision les paramètres modaux de la structure sous des conditions de vent d'amplitude courante, représentatives d'un régime de vibrations ambiantes faible en phase d'exploitation, bien loin de celles induites par des vents rares ayant une période de retour 1/50 ans. La méthodologie proposée, ainsi que le rôle central des étapes EFDD/SSI, de la calibration et de la représentation de la rigidité structurale, sont détaillés dans le corps du mémoire et structurent l'articulation entre les volets expérimental et numérique de l'étude.

Les résultats montrent que les hypothèses usuelles d'amortissement et de rigidité issues des cadres normatifs peuvent être inadaptées pour les tours élancées locales. Alors que la pratique retient couramment 2 % en conception au vent, le commentaire I du CNBC 2020 signale des valeurs in situ souvent inférieures à 1 % pour les grandes hauteurs. Un amortissement plus faible n'altère pas les fréquences propres mais accroît l'amplitude de la réponse près de la quasi-résonance, intensifiant accélérations, vitesses et déplacements. Une sensibilité qui se répercute directement sur les critères de confort et de service si l'on transpose ces valeurs dans les essais en soufflerie et les modèles de calcul. La discussion montre aussi qu'une valeur mesurée de l'amortissement de l'ordre de 0,8 % appelle à un ajustement du modèle pour reproduire fidèlement la réponse dynamique.

La stratégie adoptée, focalisée sur les paramètres réellement influents comme les modules de Young des murs et des dalles (pour calibrer la rigidité globale), puis sur des modificateurs de rigidité locale (par élément) plus sujets à des valeurs maximales non physiques, a permis une convergence cohérente des fréquences et une amélioration des corrélations modales sauf pour le troisième mode en torsion (effet de couplage : déformée modale difficilement quantifiable).

Sur le plan de la rigidité, ce mémoire met en évidence qu'une estimation biaisée de la rigidité structurelle entraîne un décalage des périodes propres, biaise les déformées modales et conduit soit à des vérifications non conservatives lorsque la rigidité est surestimée, soit à des dimensionnements excessivement conservateurs lorsqu'elle est sous-estimée. La stratégie adoptée, consiste à ajuster en priorité les paramètres physiquement dominants, tels que le module de Young des murs et des dalles qui peut variés de 80 à 120% de la valeur calculée selon l'article 8.6.2.2 et 8.6.2.3 de la norme CSA A23.3. Cette approche a permis une convergence cohérente des fréquences propres et une amélioration des déformées modales, tout en évitant des valeurs non physiques associées aux modificateurs de section fissurée. À noter qu'une perte de précision a été remarqué pour le 3$^{ème}$ mode en torsion, dû à l'effet de couplage du podium assurant la liaison avec la tour difficilement quantifiable malgré la disposition d'appareils de mesure supplémentaires dans la zone. Cette perte de précision peut aussi être dû au script Python qui a eu de la difficulté à converger aux nœuds caractérisant la déformée modale en torsion.

La procédure de calibration estimé une rigidité avec des modules d'Young passant de 31 à 37GPa dans les niveaux supérieurs à environ 45 GPa dans le podium. Localement, les modificateurs de rigidité des dalles et des murs ont convergé vers des plages physiquement significatives, comme environ 0,35 pour les dalles fissurées et environ 1,0 pour les murs en compression, validant ainsi les hypothèses normatives de la norme CSA A23.3 utilisées dans la conception. Cependant, la nécessité d'une rigidité plus élevée (jusqu'à 1,5 pour les murs et 0.55 pour les dalles) près de la base suggère une rigidité supplémentaire due à la congestion des armatures ou à des effets de transfert de niveau qui ne sont pas entièrement pris en compte dans les modèles de conception. Ces résultats soulignent la nécessité de reconsidérer les

hypothèses basées sur les codes lorsqu'une performance précise en service est requise, en particulier dans les bâtiments hauts et flexibles.

Au-delà du cas d'étude, cette recherche constitue, à l'échelle montréalaise, une première application documentée d'un protocole complet de MVA, puis d'une analyse modale avec les méthodes FFDA/ISS et enfin d'une calibration du MÉF sur une tour instrumentée de grande hauteur. Elle met en évidence l'intérêt d'un ancrage local des hypothèses de conception, fondé sur des mesures in situ locales plutôt que sur des valeurs génériques, afin d'améliorer la prédictibilité des analyses dynamiques menées au Québec. Cette étude démontre l'intérêt de combiner l'analyse modale expérimentale et la mise à jour des modèles pour les bâtiments réels, ce qui permet non seulement de valider les hypothèses de conception, mais aussi d'apporter des améliorations fondées sur des données pour les simulations futures. Bien qu'elle se limite à une seule étude de cas, cette méthodologie pourrait être reproduite pour d'autres structures canadiennes afin de constituer une base de données régionale plus précise sur les valeurs de rigidité in situ et de favoriser des pratiques de conception mieux informées.

## LIMITES

Les limites et frontières du projet reposent principalement sur la portée de l'étude et les conditions expérimentales. D'abord, l'analyse se concentre sur une seule tour de grande hauteur, ce qui ne permet pas de généraliser pleinement les résultats à d'autres typologies de bâtiments. Bien que cette étude fournisse une méthodologie préliminaire, elle ne suffit pas à établir à elle seule une méthodologie de conception et sans biais applicable à l'ensemble des bâtiments de grande hauteur aux états limites de service à Montréal.

De plus, certains étages supérieurs du bâtiment étaient encore en travaux au moment des mesures. L'absence d'enveloppe fermée a pu engendrer des courants d'air susceptibles d'amplifier localement les vibrations des planchers et ainsi d'influencer les réponses enregistrées. Ces conditions particulières peuvent donc introduire une légère incertitude dans l'interprétation des résultats in situ et doivent être prises en compte dans la discussion des conclusions.

L'intensité du vent n'a pas été mesurée pendant les MVAs, ce qui limite la capacité à dissocier l'amortissement structurel de l'effet du vent sur les valeurs identifiées.

Les MAC des déformées modales, moins sensibles que les fréquences, peuvent montrer une variabilité modérée dans le script Python, ce qui limite la précision locale de la calibration (notamment pour le 3$^{ème}$ mode en torsion).

## RECOMMANDATIONS

À la lumière de ces constats, plusieurs recommandations peuvent être formulées pour améliorer la fiabilité des analyses dynamiques et la cohérence des modèles numériques des tours en contexte montréalais :

- Intégrer systématiquement des campagnes de mesures in situ après conception (construire une base de données sur les tours montréalaise).
- Privilégier l'utilisation d'amortissements mesurés in-situ, plutôt qu'une valeur normative unique de 2 %, évitant la sous-estimation des accélérations en tête de bâtiment et des dérives inter-étages, pour :
    - La transposition des charges issues d'essais en soufflerie ;
    - Les analyses aux ÉLS.
- Ajuster la rigidité à partir des propriétés matérielles dominantes (notamment les modules de Young), améliorant la correspondance des déformées modales (MAC) et la cohérence physique du modèle mis à jour, pour :
    - Éviter de compenser la rigidité via des modificateurs dépassant des bornes physiquement plausibles.
- Constituer progressivement une base de données locale regroupant fréquences, amortissements et formes modales mesurées :
    - Faciliter la réévaluation des hypothèses normatives pour les grandes hauteurs au Canada ;
    - Affiner les fourchettes de conception selon la typologie, la hauteur et l'occupation.
- Étendre la méthodologie développée à d'autres tours (résidentielles, commerciales ou mixtes) afin de valider sa robustesse.
- Formaliser des protocoles reproductibles de mesure, de synchronisation et de traitement EFDD/SSI pour :
    - Offrir un socle commun aux firmes et laboratoires d'essais ;
    - Réduire l'écart entre hypothèses et comportements observés ;
    - Renforcer la fiabilité des vérifications de service.

# ANNEXE I

## ALGORITHME DE LA MÉTHODE EFDD

Un rappel des équations du mouvement pour une structure à plusieurs degrés de liberté est fait ci-dessous (Yun, et al. 2020) :

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) = \mathbf{F}(t) \tag{A I.1}$$

Les matrices M, C et K dans l'équation (1.1) représentent, respectivement, les matrices de masse, d'amortissement et de rigidité. Quant aux termes en fonction du temps, respectivement après la matrice M, C et K représentent l'accélération, la vitesse et les déplacements des planchers (ou d'un degré de liberté) d'un bâtiment. La force en fonction du temps, de l'autre côté de l'égalité représente la force externe appliqué à un plancher (ou degré de liberté).

Pour passer dans le domaine fréquentiel, la transformée de Fourier du déplacement **u**(t) en fonction du temps **Y**(ω) et de la force externe **F**(ω) est appliquée selon :

$$\left(-\omega^2\mathbf{M} + i\omega\mathbf{C} + \mathbf{K}\right)\mathbf{Y}(\omega) = \mathbf{F}(\omega) \tag{A I.2}$$

Comme expliqué par Yun, et al. (2020, pp. 2-3), le système linéaire avec un temps invariant est le ratio de la transformée de Fourier est le rapport entre les entrants (**Y**(ω)) et sortants (**F**(ω)), comme défini par :

$$\mathbf{Z}(\omega)\mathbf{Y}(\omega) = \mathbf{F}(\omega) \tag{A I.3}$$

L'inverse de la matrice $\mathbf{Z}(\omega)$ est exprimé comme suit par Yun, et al. (2020, p. 2) :

$$\mathbf{H}(\omega) = \mathbf{Z}^{-1}(\omega) = \frac{adj(\mathbf{Z}(\omega))}{\mathbf{Z}(\omega)} \tag{A I.4}$$

Dans cette expression, adj($\mathbf{Z}(\omega)$) est la matrice transposée de la matrice conjuguée de $\mathbf{Z}$ et $\mathbf{H}(\omega)$ la matrice de la fonction de réponse dans le domaine fréquentiel (Yun, et al., 2020, pp. 2-3). Cette expression permet notamment de construire le graphique de la densité spectrale (« PSD » en anglais). En effet, les matrices de densité spectrale sortantes $\mathbf{G}_{yy}(\omega)$ (c-à-d celles obtenus après analyse modal opérationnelle ou « OMA » en anglais) sont écrites par Yun, et al. (2020, p. 3) de la manière suivante, où $\mathbf{G}_{xx}(\omega)$ est la densité spectrale entrante (mesurée) :

$$\mathbf{G}_{yy}(\omega) = \mathbf{H}(\omega)\mathbf{G}_{xx}(\omega)\mathbf{H}(\omega)^T \tag{A I.5}$$

Après toutes simplifications et calculs faits, la forme finale de la densité spectrale sortante (graphique) prend la forme suivante (Yun, et al., 2020, p. 3) :

$$\begin{aligned}\mathbf{G}_{yy}(\omega) &\simeq \sum_{k=1}^{n} \frac{d_k \boldsymbol{\varphi}_k^* \boldsymbol{\varphi}_k^T}{i\omega - \lambda_k} + \frac{d_k \boldsymbol{\varphi}_k^* \boldsymbol{\varphi}_k^T}{-i\omega - \lambda_k^*} = \boldsymbol{\Phi}^* \left\{ diag\left[ \Re\mathrm{e}\left( \frac{2d_k}{i\omega - \lambda_k} \right) \right] \right\} \boldsymbol{\Phi}^T \\ &= \mathbf{U}\mathbf{S}\mathbf{U}^T\end{aligned} \tag{A I.6}$$

Les variables de cette expression sont définies comme étant (Yun, et al., 2020, p. 3) :

- **U** : la matrice des formes de mode unitaire où $\mathbf{U}\mathbf{U}^\mathbf{T}$ vaut la matrice identité I ;
- **S** : la matrice diagonale qui comprend les valeurs singulières des fréquences naturelles ;
- **Φ :** la matrice modale contenant les formes de mode ;
- $\mathbf{d_k}$ : le scalaire réel **;**
- $\boldsymbol{\lambda_k}$ : le $\mathbf{k}^{\mathbf{ième}}$ pole ;
- $\boldsymbol{\varphi_k}$ : le vecteur de la $k^{ième}$ forme de mode
- **i** : la valeur unitaire complexe

- **ω :** la fréquence angulaire **en rad/sec.**

Ensuite, les fréquences et périodes propres sont déterminés en sélectionnant les pics du graphique de la densité spectrale (« PSD ») (Yun, et al., 2020, p. 3).

À noter, pour plus de détails calculatoires, il convient de se référer à la référence (Yun, et al. 2020).

## ANNEXE II

## PRÉ-TRAITEMENT : SCRIPT PYTHON

Ce code permet d'éliminer le décalage temporel initial lié à la mise en place des capteurs, afin de ne conserver qu'un signal exploitable dans ARTeMIS Modal Pro.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# Traitement.py: Pré-tri des données GRILLA avant ETABS.

# "Cyrielle Seymaux"
# Copyright 2024, The 700 Building Project"

# Version 1.0.1
# "cyrielle.seymaux@polymtl.ca"


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Tromino 1
# Information initiale

DeltaT = 1/512
t0 = 0
t1 = []
i = 0

df1 = pd.read_csv("C:/Users/cysey/Documents/Mémoire/Prise de mesures -
700/Étage 40/Python - ARTeMIS/Transl. 3/Le 700 - Étage 40 Tromino 1 - Transl.
3 part1/EqualizedFile.dat", delimiter=' ')
length = len(df1)
tf = DeltaT*length

for i in range(length):
    t1.append(t0+i*DeltaT)

NS1 = df1["A"].tolist()
EW1 = df1["B"].tolist()
```

```
MaxNS1 = max(NS1)


# Tromino 2
# Information initiale

DeltaT = 1/512
t0 = 0
t2 = []
i = 0

df2 = pd.read_csv("C:/Users/cysey/Documents/Mémoire/Prise de mesures -
700/Étage 40/Python - ARTeMIS/Transl. 3/Le 700 - Étage 40 Tromino 2 - Transl.
3 part1/EqualizedFile.dat", delimiter=' ')
length = len(df2)
tf = DeltaT*length

for i in range(length):
    t2.append(t0+i*DeltaT)

NS2 = df2["A"].tolist()
EW2 = df2["B"].tolist()

MaxNS2 = max(NS2)


# Tromino 3
# Information initiale

DeltaT = 1/512
t0 = 0
t3 = []
i = 0

df3 = pd.read_csv("C:/Users/cysey/Documents/Mémoire/Prise de mesures -
700/Étage 40/Python - ARTeMIS/Transl. 3/Le 700 - Étage 40 Tromino 3 - Transl.
3 part1/EqualizedFile.dat", delimiter=' ')
length = len(df3)
tf = DeltaT*length

for i in range(length):
    t3.append(t0+i*DeltaT)

NS3 = df3["A"].tolist()
```

```
EW3 = df3["B"].tolist()

MaxNS3 = max(NS3)


# Tromino 4
# Information initiale

DeltaT = 1/512
t0 = 0
t4 = []
i = 0

df4 = pd.read_csv("C:/Users/cysey/Documents/Mémoire/Prise de mesures -
700/Étage 40/Python - ARTeMIS/Transl. 3/Le 700 - Étage 40 Tromino 4 - Transl.
3 part1/EqualizedFile.dat", delimiter=' ')
length = len(df4)
tf = DeltaT*length

for i in range(length):
    t4.append(t0+i*DeltaT)

NS4 = df4["A"].tolist()
EW4 = df4["B"].tolist()

MaxNS4 = max(NS4)


# Tromino bleu - réf.
# Information initiale

DeltaT = 1/512
t0 = 0
tref = []
i = 0

dfref = pd.read_csv("C:/Users/cysey/Documents/Mémoire/Prise de mesures -
700/Étage 40/Python - ARTeMIS/Transl. 3/Le 700 - Étage 40 Tromino b - Transl.
3 part1/EqualizedFile.dat", delimiter=' ')

length = len(dfref)
tf = DeltaT*length

for i in range(length):
```

```
    tref.append(t0+i*DeltaT)

NSref = dfref["A"].tolist()
EWref = dfref["B"].tolist()

MaxNSref = max(NSref)

# Graphiques NS
plt.plot(t1, NS1, label = "Tromino 1")
plt.plot(t2, NS2, label = "Tromino 2")
plt.plot(t3, NS3, label = "Tromino 3")
plt.plot(t4, NS4, label = "Tromino 4")
plt.plot(tref, NSref, label = "Référence")
plt.ylabel("Vitesse en mm/s")
plt.xlabel("Temps en secondes")
plt.title("Initial time history - NS direction")
plt.legend()
plt.show()


# Graphique EW
plt.plot(t1, EW1, label = "Tromino 1")
plt.plot(t2, EW2, label = "Tromino 2")
plt.plot(t3, EW3, label = "Tromino 3")
plt.plot(t4, EW4, label = "Tromino 4")
plt.plot(tref, EWref, label = "Référence")
plt.ylabel("Vitesse en mm/s")
plt.xlabel("Temps en secondes")
plt.title("Initial time history - EW direction")
plt.legend()
plt.show()


# DeltaT à enlever

# Hypothèse 8min45

Time = 8*60+45 # secondes
Nbpoints = int(Time/DeltaT+1)+1
Tf = tref[Nbpoints:]


df1 = df1[Nbpoints:]
NS1 = df1["A"].tolist()
```

```
EW1 = df1["B"].tolist()

df2 = df2[Nbpoints:]
NS2 = df2["A"].tolist()
EW2 = df2["B"].tolist()

df3 = df3[Nbpoints:]
NS3 = df3["A"].tolist()
EW3 = df3["B"].tolist()

df4 = df4[Nbpoints:]
NS4 = df4["A"].tolist()
EW4 = df4["B"].tolist()

dfref = dfref[Nbpoints:]
NSref = dfref["A"].tolist()
EWref = dfref["B"].tolist()


# Graphiques NS - final
plt.plot(Tf, NSref, label = "Référence")
plt.plot(Tf, NS1, label = "Tromino 1")
plt.plot(Tf, NS2, label = "Tromino 2")
plt.plot(Tf, NS3, label = "Tromino 3")
plt.plot(Tf, NS4, label = "Tromino 4")

plt.ylabel("Vitesse en mm/s")
plt.xlabel("Temps en secondes")
plt.title("Final time history - NS direction")
plt.legend()
plt.show()

# Graphique EW
plt.plot(Tf, EWref, label = "Référence")
plt.plot(Tf, EW1, label = "Tromino 1")
plt.plot(Tf, EW2, label = "Tromino 2")
plt.plot(Tf, EW3, label = "Tromino 3")
plt.plot(Tf, EW4, label = "Tromino 4")

plt.ylabel("Vitesse en mm/s")
plt.xlabel("Temps en secondes")
plt.title("Final time history - EW direction")
plt.legend()
plt.show()
```

```
# Exportation ARTeMIS

ARTeMISdf = pd.DataFrame()
list_of_tuples = list(zip(NS1, EW1, NS2, EW2, NS3, EW3, NS4, EW4, NSref,
EWref))
ARTeMISdf = pd.DataFrame(list_of_tuples,columns=['NS1', 'EW1', "NS2",
                                                 "EW2", "NS3", "EW3",
                                                 "NS4", "EW4", "NSref",
                                                 "EWref"])

string_representation = ARTeMISdf.applymap(lambda x: f"   {x:.7e}" if x >= 0
else f"  {x:.7e}").apply(lambda x: ''.join(x), axis=1)

# Écrire la chaîne de caractères dans un fichier texte
with open('Artemis_input.txt', 'w') as f:
    f.write('\n'.join(string_representation))

# FIN
```

## ANNEXE III

## ANALYSE DE SENSIBILITÉ GLOBALE : SCRIPT PYTHON

Ce script permet de conduire une analyse de sensibilité des paramètres physiques du modèle numérique de la tour, en utilisant l'API d'ETABS.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# test.py : Analyse de sensibilité globle.

# "Cyrielle Seymaux"
# Copyright 2024, The 700 Building Project"

# Version 1.0.1
# "cyrielle.seymaux@polymtl.ca"


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import FancyArrowPatch
import seaborn as sns
from matplotlib.colors import LinearSegmentedColormap

# ------------------------------------------------
# ZONE 1
# Charger le fichier Excel
file_path = r"z:/Cyrielle/4. Mémoire\Analyse de sensibilité/FINAL FINAL ZONE
1.xlsx"
xls = pd.ExcelFile(file_path)

# Créer un dictionnaire pour stocker les DataFrames par feuille
dfs_modes = {}

# Plage de lignes à extraire (A7 à A41)
start_row = 5
end_row = 40
param_col = 0
value_col = 3
```

```
# Lire chaque feuille et extraire les données souhaitées
for sheet in xls.sheet_names:
    df = pd.read_excel(xls, sheet_name=sheet)
    df_subset = df.iloc[start_row:end_row, [param_col, value_col]].copy()
    df_subset.columns = ["Paramètre", "Valeur"]
    df_subset.dropna(inplace=True)
    dfs_modes[sheet] = df_subset

# Exemple : afficher la feuille "freq_Freq_1"
print(dfs_modes["freq_Freq_1"])

df_mode1 = dfs_modes["freq_Freq_1"]
df_mode2 = dfs_modes["freq_Freq_2"]
df_mode3 = dfs_modes["freq_Freq_3"]
df_mode4 = dfs_modes["freq_Freq_4"]
df_mode5 = dfs_modes["freq_Freq_5"]
df_mode6 = dfs_modes["freq_Freq_6"]
df_mode7 = dfs_modes["freq_Freq_7"]
df_mode8 = dfs_modes["freq_Freq_8"]
df_mode9 = dfs_modes["freq_Freq_9"]
df_mode10 = dfs_modes["freq_Freq_10"]
df_mode11 = dfs_modes["freq_Freq_11"]
df_mode12 = dfs_modes["freq_Freq_12"]
df_mode13 = dfs_modes["freq_Freq_13"]
df_mode14 = dfs_modes["freq_Freq_14"]

# Convertir en tableaux numpy
ModuleYoungCol_pct = []
ModuleYoungBeamOrCoupled_pct = []
ModuleYoungBeamCoupled_pct = []
Ix_beam_pct = []
Iy_beam_pct = []
Ix_col_pct = []
Iy_col_pct = []
Ix_beamC_pct = []
Iy_beamC_pct = []
Jbc_pct = []
Jb_pct = []
Jc_pct = []
ModuleYoungSlab_pct = []
f11s_pct = []
f22s_pct = []
f12s_pct = []
m11s_pct = []
```

```
m22s_pct = []
m12s_pct = []
ModuleYoungWall_pct = []
f11w_pct = []
f22w_pct = []
f12w_pct = []

for i in range(14):
    ModuleYoungCol_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][5])*10
0)
    ModuleYoungBeamOrCoupled_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeu
r"][6])*100)
    ModuleYoungBeamCoupled_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"
][7])*100)
    Ix_beam_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][8])*100)
    Iy_beam_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][9])*100)
    Ix_col_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][10])*100)
    Iy_col_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][11])*100)
    Ix_beamC_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][12])*100)
    Iy_beamC_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][13])*100)
    Jbc_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][14])*100)
    Jb_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][15])*100)
    Jc_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][16])*100)
    ModuleYoungSlab_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][17])*
100)
    f11s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][18])*100)
    f22s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][19])*100)
    f12s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][20])*100)
    m11s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][21])*100)
    m22s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][22])*100)
    m12s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][23])*100)
    ModuleYoungWall_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][24])*
100)
    f11w_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][25])*100)
    f22w_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][26])*100)
    f12w_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][27])*100)

data_matrix = np.array([
    ModuleYoungCol_pct,
    ModuleYoungBeamOrCoupled_pct,
    ModuleYoungBeamCoupled_pct,
    ModuleYoungSlab_pct,
    ModuleYoungWall_pct,
    Ix_beam_pct,
```

```
    Iy_beam_pct,
    Ix_col_pct,
    Iy_col_pct,
    Ix_beamC_pct,
    Iy_beamC_pct,
    Jbc_pct,
    Jb_pct,
    Jc_pct,
    f11s_pct, f22s_pct, f12s_pct,
    m11s_pct, m22s_pct, m12s_pct,
    f11w_pct, f22w_pct, f12w_pct
]).T  # shape: (14 modes x n paramètres)

# Données de base
n_modes = 5
modes = [f"Mode {i+1}" for i in range(n_modes)]

# Paramètres à inclure dans la heatmap (liste complète basée sur les noms des
listes existantes)
param_data = {
    "Column Young’s modulus": ModuleYoungCol_pct,
    "Beam Young’s modulus\n(excl. coupling beams)":
ModuleYoungBeamOrCoupled_pct,
    "Coupling Beam Young’s modulus": ModuleYoungBeamCoupled_pct,
    "Slab Young’s modulus": ModuleYoungSlab_pct,
    "Wall Young’s modulus": ModuleYoungWall_pct,
    #"Ix_b": Ix_beam_pct,
    "Column inertia in x": Ix_col_pct,
    "Column inertia in y": Iy_col_pct,
    "Beam inertia in y\n(excl. coupling beams)": Iy_beam_pct,
    #"Ix_bC": Ix_beamC_pct,
    "Coupling Beam inertia in y": Iy_beamC_pct,
    #"J_bC": Jbc_pct,
    #"f11_s": f11s_pct,
    #"f22_s": f22s_pct,
    #"f12_s": f12s_pct,
    "Slab out-plane stiffness (m11)": m11s_pct,
    "Slab out-plane stiffness (m22)": m22s_pct,
    "Slab out-plane stiffness (m12)": m12s_pct,
    "Wall in-plane stiffness (f11)": f11w_pct,
    "Wall in-plane stiffness (f22)": f22w_pct,
    "Wall in-plane stiffness (f12)": f12w_pct,
    #"Beam torsional Constant\n(excl. coupling beams)": Jb_pct
}
```

```
# Construction de la matrice de données
param_labels = list(param_data.keys())
data_matrix = np.array([param_data[param][:n_modes] for param in
param_labels]).T  # shape: (9 modes x N paramètres)
 # shape: (14 modes x N paramètres)

# Palette personnalisée
colors = [
    (0.0, "#ff9999"),  # 0% → rouge pâle (départ)
    (0.3, "#ff6666"),  # 30% → rouge moyen
    (1.0, "#cc0000")   # 100% → rouge foncé
]
custom_cmap = LinearSegmentedColormap.from_list("custom_reds", colors, N=256)

# Création de la heatmap
plt.figure(figsize=(6, 15))
ax = sns.heatmap(
    data_matrix.T,  # <--- transpose ici
    cmap=custom_cmap,
    xticklabels=modes,          # ← maintenant sur l'axe X
    yticklabels=param_labels,   # ← maintenant sur l'axe Y
    annot=False,
    linewidths=0.5,
    linecolor='white'
)

# Récupérer la colorbar et modifier la taille du label et des ticks
cbar = ax.collections[0].colorbar
cbar.ax.tick_params(labelsize=15)


# Ajout manuel des annotations dans chaque cellule
for i in range(data_matrix.shape[1]):  # ← nombre de paramètres (lignes
maintenant)
    for j in range(data_matrix.shape[0]):  # ← nombre de modes (colonnes
maintenant)
        value = data_matrix[j, i]  # ← on accède à la valeur originale, non
transposée
        ax.text(j + 0.5, i + 0.5, f"{value:.2f}%",
                ha='center', va='center', color='black', fontsize=12)

# Tracer des lignes de séparation entre groupes de paramètres (axes inversés
= lignes)
```

```
# Les indices correspondent aux Y des lignes (entre les lignes, donc 1.5,
2.5, etc.)

# Liste des positions de séparation (lignes horizontales)
separators = [5, 9, 12, 15]

# Tracer les lignes
for sep in separators:
    # Ligne pleine dans la heatmap (zone des données uniquement)
    ax.hlines(sep, 0, len(modes), colors='black', linewidth=1.2)

    # Ligne pointillée juste à gauche de la 1ère colonne, sans changer les
axes
    ax.annotate(
        '',
        xy=(0, sep),
        xytext=(-2.7, sep),  # ← ajustable si tu veux plus long
        xycoords='data',
        textcoords='data',
        arrowprops=dict(
            arrowstyle='-',
            linestyle='dashed',
            color='black',
            linewidth=1.2
        )
    )

# Taille des ticks (étiquettes sur les axes X et Y)
ax.tick_params(axis='x', labelsize=12)
ax.tick_params(axis='y', labelsize=12)


plt.tight_layout()
plt.show()
plt.savefig("figure_for_word.png", dpi=300, bbox_inches='tight')

# ----------------------------------------------------------------------------
----------------------------------
# ZONE 2
# Charger le fichier Excel
file_path = r"z:/Cyrielle/4. Mémoire\Analyse de sensibilité/FINAL FINAL ZONE
2.xlsx"
xls = pd.ExcelFile(file_path)
```

```
# Créer un dictionnaire pour stocker les DataFrames par feuille
dfs_modes = {}

# Plage de lignes à extraire (A7 à A41)
start_row = 5
end_row = 40
param_col = 0
value_col = 3

# Lire chaque feuille et extraire les données souhaitées
for sheet in xls.sheet_names:
    df = pd.read_excel(xls, sheet_name=sheet)
    df_subset = df.iloc[start_row:end_row, [param_col, value_col]].copy()
    df_subset.columns = ["Paramètre", "Valeur"]
    df_subset.dropna(inplace=True)
    dfs_modes[sheet] = df_subset

# Exemple : afficher la feuille "freq_Freq_1"
print(dfs_modes["freq_Freq_1"])

df_mode1 = dfs_modes["freq_Freq_1"]
df_mode2 = dfs_modes["freq_Freq_2"]
df_mode3 = dfs_modes["freq_Freq_3"]
df_mode4 = dfs_modes["freq_Freq_4"]
df_mode5 = dfs_modes["freq_Freq_5"]
df_mode6 = dfs_modes["freq_Freq_6"]
df_mode7 = dfs_modes["freq_Freq_7"]
df_mode8 = dfs_modes["freq_Freq_8"]
df_mode9 = dfs_modes["freq_Freq_9"]
df_mode10 = dfs_modes["freq_Freq_10"]
df_mode11 = dfs_modes["freq_Freq_11"]
df_mode12 = dfs_modes["freq_Freq_12"]
df_mode13 = dfs_modes["freq_Freq_13"]
df_mode14 = dfs_modes["freq_Freq_14"]

# Convertir en tableaux numpy
ModuleYoungCol_pct = []
ModuleYoungBeamOrCoupled_pct = []
ModuleYoungBeamCoupled_pct = []
Ix_beam_pct = []
Iy_beam_pct = []
Ix_col_pct = []
Iy_col_pct = []
Ix_beamC_pct = []
```

```
Iy_beamC_pct = []
Jbc_pct = []
Jb_pct = []
Jc_pct = []
ModuleYoungSlab_pct = []
f11s_pct = []
f22s_pct = []
f12s_pct = []
m11s_pct = []
m22s_pct = []
m12s_pct = []
ModuleYoungWall_pct = []
f11w_pct = []
f22w_pct = []
f12w_pct = []

for i in range(14):
    ModuleYoungCol_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][5])*10
0)
    ModuleYoungBeamOrCoupled_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeu
r"][6])*100)
    ModuleYoungBeamCoupled_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"
][7])*100)
    Ix_beam_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][8])*100)
    Iy_beam_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][9])*100)
    Ix_col_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][10])*100)
    Iy_col_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][11])*100)
    Ix_beamC_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][12])*100)
    Iy_beamC_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][13])*100)
    Jbc_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][14])*100)
    Jb_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][15])*100)
    Jc_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][16])*100)
    ModuleYoungSlab_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][17])*
100)
    f11s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][18])*100)
    f22s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][19])*100)
    f12s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][20])*100)
    m11s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][21])*100)
    m22s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][22])*100)
    m12s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][23])*100)
    ModuleYoungWall_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][24])*
100)
    f11w_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][25])*100)
    f22w_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][26])*100)
```

```
    f12w_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][27])*100)

data_matrix = np.array([
    ModuleYoungCol_pct,
    ModuleYoungBeamOrCoupled_pct,
    ModuleYoungBeamCoupled_pct,
    ModuleYoungSlab_pct,
    ModuleYoungWall_pct,
    Ix_beam_pct,
    Iy_beam_pct,
    Ix_col_pct,
    Iy_col_pct,
    Ix_beamC_pct,
    Iy_beamC_pct,
    Jbc_pct,
    Jb_pct,
    Jc_pct,
    f11s_pct, f22s_pct, f12s_pct,
    m11s_pct, m22s_pct, m12s_pct,
    f11w_pct, f22w_pct, f12w_pct
]).T  # shape: (14 modes x n paramètres)

# Données de base
n_modes = 5
modes = [f"Mode {i+1}" for i in range(n_modes)]

# Paramètres à inclure dans la heatmap (liste complète basée sur les noms des
listes existantes)
param_data = {
    "Column Young’s modulus": ModuleYoungCol_pct,
    "Beam Young’s modulus\n(excl. coupling beams)":
ModuleYoungBeamOrCoupled_pct,
    "Coupling Beam Young’s modulus": ModuleYoungBeamCoupled_pct,
    "Slab Young’s modulus": ModuleYoungSlab_pct,
    "Wall Young’s modulus": ModuleYoungWall_pct,
    #"Ix_b": Ix_beam_pct,
    "Column inertia in x": Ix_col_pct,
    "Column inertia in y": Iy_col_pct,
    "Beam inertia in y\n(excl. coupling beams)": Iy_beam_pct,
    #"Ix_bC": Ix_beamC_pct,
    "Coupling Beam inertia in y": Iy_beamC_pct,
    #"J_bC": Jbc_pct,
    #"f11_s": f11s_pct,
    #"f22_s": f22s_pct,
```

```
    #"f12_s": f12s_pct,
    "Slab out-plane stiffness (m11)": m11s_pct,
    "Slab out-plane stiffness (m22)": m22s_pct,
    "Slab out-plane stiffness (m12)": m12s_pct,
    "Wall in-plane stiffness (f11)": f11w_pct,
    "Wall in-plane stiffness (f22)": f22w_pct,
    "Wall in-plane stiffness (f12)": f12w_pct,
    #"Beam torsional Constant\n(excl. coupling beams)": Jb_pct
}

# Construction de la matrice de données
param_labels = list(param_data.keys())
data_matrix = np.array([param_data[param][:n_modes] for param in
param_labels]).T   # shape: (14 modes x N paramètres)

# Palette personnalisée
colors = [
    (0.0, "#ffd199"),  # orange pâle
    (0.3, "#ffa500"),  # orange (pur)
    (1.0, "#cc7000")   # orange foncé/brûlé
]
custom_cmap = LinearSegmentedColormap.from_list("custom_reds", colors, N=256)

# Création de la heatmap
plt.figure(figsize=(6, 15))
ax = sns.heatmap(
    data_matrix.T,  # <--- transpose ici
    cmap=custom_cmap,
    xticklabels=modes,          # ← maintenant sur l'axe X
    yticklabels=param_labels,   # ← maintenant sur l'axe Y
    cbar_kws={'label': 'Effet moyen μ* du paramètre sur la réponse modale
(%)'},
    annot=False,
    linewidths=0.5,
    linecolor='white'
)

# Récupérer la colorbar et modifier la taille du label et des ticks
cbar = ax.collections[0].colorbar
cbar.ax.tick_params(labelsize=15)


# Ajout manuel des annotations dans chaque cellule
```

```
for i in range(data_matrix.shape[1]):  # ← nombre de paramètres (lignes
maintenant)
    for j in range(data_matrix.shape[0]):  # ← nombre de modes (colonnes
maintenant)
        value = data_matrix[j, i]  # ← on accède à la valeur originale, non
transposée
        ax.text(j + 0.5, i + 0.5, f"{value:.2f}%",
                ha='center', va='center', color='black', fontsize=12)

# Tracer des lignes de séparation entre groupes de paramètres (axes inversés
= lignes)
# Les indices correspondent aux Y des lignes (entre les lignes, donc 1.5,
2.5, etc.)

# Liste des positions de séparation (lignes horizontales)
separators = [5, 9, 12, 15]

# Tracer les lignes
for sep in separators:
    # Ligne pleine dans la heatmap (zone des données uniquement)
    ax.hlines(sep, 0, len(modes), colors='black', linewidth=1.2)

    # Ligne pointillée juste à gauche de la 1ère colonne, sans changer les
axes
    ax.annotate(
        '',
        xy=(0, sep),
        xytext=(-2.7, sep),  # ← ajustable si tu veux plus long
        xycoords='data',
        textcoords='data',
        arrowprops=dict(
            arrowstyle='-',
            linestyle='dashed',
            color='black',
            linewidth=1.2
        )
    )

# Taille des ticks (étiquettes sur les axes X et Y)
ax.tick_params(axis='x', labelsize=12)
ax.tick_params(axis='y', labelsize=12)


plt.tight_layout()
```

```
plt.show()
plt.savefig("figure_for_word.png", dpi=300, bbox_inches='tight')


# ------------------------------------------------------------------------
----------------------------------
# ZONE 3
# Charger le fichier Excel
file_path = r"z:/Cyrielle/4. Mémoire\Analyse de sensibilité/FINAL FINAL ZONE
3.xlsx"
xls = pd.ExcelFile(file_path)

# Créer un dictionnaire pour stocker les DataFrames par feuille
dfs_modes = {}

# Plage de lignes à extraire (A7 à A41)
start_row = 5
end_row = 40
param_col = 0
value_col = 3

# Lire chaque feuille et extraire les données souhaitées
for sheet in xls.sheet_names:
    df = pd.read_excel(xls, sheet_name=sheet)
    df_subset = df.iloc[start_row:end_row, [param_col, value_col]].copy()
    df_subset.columns = ["Paramètre", "Valeur"]
    df_subset.dropna(inplace=True)
    dfs_modes[sheet] = df_subset

# Exemple : afficher la feuille "freq_Freq_1"
print(dfs_modes["freq_Freq_1"])

df_mode1 = dfs_modes["freq_Freq_1"]
df_mode2 = dfs_modes["freq_Freq_2"]
df_mode3 = dfs_modes["freq_Freq_3"]
df_mode4 = dfs_modes["freq_Freq_4"]
df_mode5 = dfs_modes["freq_Freq_5"]
df_mode6 = dfs_modes["freq_Freq_6"]
df_mode7 = dfs_modes["freq_Freq_7"]
df_mode8 = dfs_modes["freq_Freq_8"]
df_mode9 = dfs_modes["freq_Freq_9"]
df_mode10 = dfs_modes["freq_Freq_10"]
df_mode11 = dfs_modes["freq_Freq_11"]
df_mode12 = dfs_modes["freq_Freq_12"]
```

```
df_mode13 = dfs_modes["freq_Freq_13"]
df_mode14 = dfs_modes["freq_Freq_14"]

# Convertir en tableaux numpy
ModuleYoungCol_pct = []
ModuleYoungBeamOrCoupled_pct = []
ModuleYoungBeamCoupled_pct = []
Ix_beam_pct = []
Iy_beam_pct = []
Ix_col_pct = []
Iy_col_pct = []
Ix_beamC_pct = []
Iy_beamC_pct = []
Jbc_pct = []
Densitebc_pct = []
Airebc_pct = []
Jb_pct = []
Densiteb_pct = []
Aireb_pct = []
Jc_pct = []
Densitec_pct = []
Airec_pct = []
ModuleYoungSlab_pct = []
Densites_pct = []
Thickness250_s_pct = []
Thickness225_s_pct = []
f11s_pct = []
f22s_pct = []
f12s_pct = []
m11s_pct = []
m22s_pct = []
m12s_pct = []
ModuleYoungWall_pct = []
ModuleYoungWall1_pct = []
Densitew_pct = []
f11w_pct = []
f22w_pct = []
f12w_pct = []
f11w1_pct = []
f22w1_pct = []
f12w1_pct = []

for i in range(14):
```

```
    ModuleYoungCol_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][5])*10
0)
    ModuleYoungBeamOrCoupled_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeu
r"][6])*100)
    ModuleYoungBeamCoupled_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"
][7])*100)
    Ix_beam_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][8])*100)
    Iy_beam_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][9])*100)
    Ix_col_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][10])*100)
    Iy_col_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][11])*100)
    Ix_beamC_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][12])*100)
    Iy_beamC_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][13])*100)
    Jbc_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][14])*100)
    Jb_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][15])*100)
    Jc_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][16])*100)
    ModuleYoungSlab_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][17])*
100)
    f11s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][18])*100)
    f22s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][19])*100)
    f12s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][20])*100)
    m11s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][21])*100)
    m22s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][22])*100)
    m12s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][23])*100)
    ModuleYoungWall_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][24])*
100)
    ModuleYoungWall1_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][25])
*100)
    f11w_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][26])*100)
    f22w_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][27])*100)
    f12w_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][28])*100)
    f11w1_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][29])*100)
    f22w1_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][30])*100)
    f12w1_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][31])*100)

data_matrix = np.array([
    ModuleYoungCol_pct,
    ModuleYoungBeamOrCoupled_pct,
    ModuleYoungBeamCoupled_pct,
    ModuleYoungSlab_pct,
    ModuleYoungWall_pct,
    ModuleYoungWall1_pct,
    Ix_beam_pct,
    Iy_beam_pct,
    Ix_col_pct,
```

```
    Iy_col_pct,
    Ix_beamC_pct,
    Iy_beamC_pct,
    Jbc_pct,
    Jb_pct,
    Jc_pct,
    f11s_pct, f22s_pct, f12s_pct,
    m11s_pct, m22s_pct, m12s_pct,
    f11w_pct, f22w_pct, f12w_pct,
    f11w1_pct, f22w1_pct, f12w1_pct
]).T  # shape: (14 modes x n paramètres)

# Données de base
n_modes = 5
modes = [f"Mode {i+1}" for i in range(n_modes)]

# Paramètres à inclure dans la heatmap (liste complète basée sur les noms des
listes existantes)
param_data = {
    "Column Young’s modulus": ModuleYoungCol_pct,
    "Beam Young’s modulus\n(excl. coupling beams)":
ModuleYoungBeamOrCoupled_pct,
    "Coupling Beam Young’s modulus": ModuleYoungBeamCoupled_pct,
    "Slab Young’s modulus": ModuleYoungSlab_pct,
    "Wall Young’s modulus": [x + y for x, y in zip(ModuleYoungWall_pct,
ModuleYoungWall1_pct)],
    #"Ix_b": Ix_beam_pct,
    "Column inertia in x": Ix_col_pct,
    "Column inertia in y": Iy_col_pct,
    "Beam inertia in y\n(excl. coupling beams)": Iy_beam_pct,
    #"Ix_bC": Ix_beamC_pct,
    "Coupling Beam inertia in y": Iy_beamC_pct,
    #"J_bC": Jbc_pct,
    #"f11_s": f11s_pct,
    #"f22_s": f22s_pct,
    #"f12_s": f12s_pct,
    "Slab out-plane stiffness (m11)": m11s_pct,
    "Slab out-plane stiffness (m22)": m22s_pct,
    "Slab out-plane stiffness (m12)": m12s_pct,
    "Wall in-plane stiffness (f11)": [x + y for x, y in zip(f11w_pct,
f11w1_pct)],
    "Wall in-plane stiffness (f22)": [x + y for x, y in zip(f22w_pct,
f22w1_pct)],
```

```
    "Wall in-plane stiffness (f12)": [x + y for x, y in zip(f12w_pct,
f12w1_pct)]
    #"Beam torsional Constant\n(excl. coupling beams)": Jb_pct
}

# Construction de la matrice de données
param_labels = list(param_data.keys())
data_matrix = np.array([param_data[param][:n_modes] for param in
param_labels]).T   # shape: (14 modes x N paramètres)

# Palette personnalisée
colors = [
    (0.0, "#fff7cc"),  # jaune très pâle (pastel, crème)
    (0.3, "#ffe680"),  # jaune doux (style post-it pâle)
    (1.0, "#e6b800")   # jaune foncé/doré élégant
]
custom_cmap = LinearSegmentedColormap.from_list("custom_reds", colors, N=256)

# Création de la heatmap
plt.figure(figsize=(6, 15))
ax = sns.heatmap(
    data_matrix.T,  # <--- transpose ici
    cmap=custom_cmap,
    xticklabels=modes,          # ← maintenant sur l'axe X
    yticklabels=param_labels,   # ← maintenant sur l'axe Y
    cbar_kws={'label': 'Effet moyen μ* du paramètre sur la réponse modale
(%)'},
    annot=False,
    linewidths=0.5,
    linecolor='white'
)

# Récupérer la colorbar et modifier la taille du label et des ticks
cbar = ax.collections[0].colorbar
cbar.ax.tick_params(labelsize=15)


# Ajout manuel des annotations dans chaque cellule
for i in range(data_matrix.shape[1]):  # ← nombre de paramètres (lignes
maintenant)
    for j in range(data_matrix.shape[0]):  # ← nombre de modes (colonnes
maintenant)
        value = data_matrix[j, i]  # ← on accède à la valeur originale, non
transposée
```

```
        ax.text(j + 0.5, i + 0.5, f"{value:.2f}%",
                ha='center', va='center', color='black', fontsize=12)

# Tracer des lignes de séparation entre groupes de paramètres (axes inversés
= lignes)
# Les indices correspondent aux Y des lignes (entre les lignes, donc 1.5,
2.5, etc.)

# Liste des positions de séparation (lignes horizontales)
separators = [5, 9, 12, 15]

# Tracer les lignes
for sep in separators:
    # Ligne pleine dans la heatmap (zone des données uniquement)
    ax.hlines(sep, 0, len(modes), colors='black', linewidth=1.2)

    # Ligne pointillée juste à gauche de la 1ère colonne, sans changer les
axes
    ax.annotate(
        '',
        xy=(0, sep),
        xytext=(-2.7, sep),  # ← ajustable si tu veux plus long
        xycoords='data',
        textcoords='data',
        arrowprops=dict(
            arrowstyle='-',
            linestyle='dashed',
            color='black',
            linewidth=1.2
        )
    )

# Taille des ticks (étiquettes sur les axes X et Y)
ax.tick_params(axis='x', labelsize=12)
ax.tick_params(axis='y', labelsize=12)


plt.tight_layout()
plt.show()
plt.savefig("figure_for_word.png", dpi=300, bbox_inches='tight')


# ----------------------------------------------------------------------
----------------------------------
```

```
# ZONE 4
# Charger le fichier Excel
file_path = r"z:/Cyrielle/4. Mémoire\Analyse de sensibilité/FINAL FINAL ZONE
4.xlsx"
xls = pd.ExcelFile(file_path)

# Créer un dictionnaire pour stocker les DataFrames par feuille
dfs_modes = {}

# Plage de lignes à extraire (A7 à A41)
start_row = 5
end_row = 40
param_col = 0
value_col = 3

# Lire chaque feuille et extraire les données souhaitées
for sheet in xls.sheet_names:
    df = pd.read_excel(xls, sheet_name=sheet)
    df_subset = df.iloc[start_row:end_row, [param_col, value_col]].copy()
    df_subset.columns = ["Paramètre", "Valeur"]
    df_subset.dropna(inplace=True)
    dfs_modes[sheet] = df_subset

# Exemple : afficher la feuille "freq_Freq_1"
print(dfs_modes["freq_Freq_1"])

df_mode1 = dfs_modes["freq_Freq_1"]
df_mode2 = dfs_modes["freq_Freq_2"]
df_mode3 = dfs_modes["freq_Freq_3"]
df_mode4 = dfs_modes["freq_Freq_4"]
df_mode5 = dfs_modes["freq_Freq_5"]
df_mode6 = dfs_modes["freq_Freq_6"]
df_mode7 = dfs_modes["freq_Freq_7"]
df_mode8 = dfs_modes["freq_Freq_8"]
df_mode9 = dfs_modes["freq_Freq_9"]
df_mode10 = dfs_modes["freq_Freq_10"]
df_mode11 = dfs_modes["freq_Freq_11"]
df_mode12 = dfs_modes["freq_Freq_12"]
df_mode13 = dfs_modes["freq_Freq_13"]
df_mode14 = dfs_modes["freq_Freq_14"]

# Convertir en tableaux numpy
ModuleYoungCol_pct = []
ModuleYoungBeamOrCoupled_pct = []
```

```
ModuleYoungBeamCoupled_pct = []
Ix_beam_pct = []
Iy_beam_pct = []
Ix_col_pct = []
Iy_col_pct = []
Ix_beamC_pct = []
Iy_beamC_pct = []
Jbc_pct = []
Densitebc_pct = []
Airebc_pct = []
Jb_pct = []
Densiteb_pct = []
Aireb_pct = []
Jc_pct = []
Densitec_pct = []
Airec_pct = []
ModuleYoungSlab_pct = []
Densites_pct = []
Thickness250_s_pct = []
Thickness225_s_pct = []
f11s_pct = []
f22s_pct = []
f12s_pct = []
m11s_pct = []
m22s_pct = []
m12s_pct = []
ModuleYoungWall_pct = []
ModuleYoungWall1_pct = []
Densitew_pct = []
f11w_pct = []
f22w_pct = []
f12w_pct = []
f11w1_pct = []
f22w1_pct = []
f12w1_pct = []

for i in range(14):
    ModuleYoungCol_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][5])*10
0)
    ModuleYoungBeamOrCoupled_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeu
r"][6])*100)
    ModuleYoungBeamCoupled_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"
][7])*100)
    Ix_beam_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][8])*100)
```

```
    Iy_beam_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][9])*100)
    Ix_col_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][10])*100)
    Iy_col_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][11])*100)
    Ix_beamC_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][12])*100)
    Iy_beamC_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][13])*100)
    Jbc_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][14])*100)
    Jb_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][15])*100)
    Jc_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][16])*100)
    ModuleYoungSlab_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][17])*
100)
    f11s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][18])*100)
    f22s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][19])*100)
    f12s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][20])*100)
    m11s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][21])*100)
    m22s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][22])*100)
    m12s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][23])*100)
    ModuleYoungWall_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][24])*
100)
    f11w_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][25])*100)
    f22w_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][26])*100)
    f12w_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][27])*100)
    f11w1_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][28])*100)
    f22w1_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][29])*100)
    f12w1_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][30])*100)

data_matrix = np.array([
    ModuleYoungCol_pct,
    ModuleYoungBeamOrCoupled_pct,
    ModuleYoungBeamCoupled_pct,
    ModuleYoungSlab_pct,
    ModuleYoungWall_pct,
    Ix_beam_pct,
    Iy_beam_pct,
    Ix_col_pct,
    Iy_col_pct,
    Ix_beamC_pct,
    Iy_beamC_pct,
    Jbc_pct,
    Jb_pct,
    Jc_pct,
    f11s_pct, f22s_pct, f12s_pct,
    m11s_pct, m22s_pct, m12s_pct,
    f11w_pct, f22w_pct, f12w_pct,
    f11w1_pct, f22w1_pct, f12w1_pct
```

```
]).T  # shape: (14 modes x n paramètres)

# Données de base
n_modes = 5
modes = [f"Mode {i+1}" for i in range(n_modes)]

# Paramètres à inclure dans la heatmap (liste complète basée sur les noms des
listes existantes)
param_data = {
    "Column Young’s modulus": ModuleYoungCol_pct,
    "Beam Young’s modulus\n(excl. coupling beams)":
ModuleYoungBeamOrCoupled_pct,
    "Coupling Beam Young’s modulus": ModuleYoungBeamCoupled_pct,
    "Slab Young’s modulus": ModuleYoungSlab_pct,
    "Wall Young’s modulus": ModuleYoungWall_pct,
    #"Ix_b": Ix_beam_pct,
    "Column inertia in x": Ix_col_pct,
    "Column inertia in y": Iy_col_pct,
    "Beam inertia in y\n(excl. coupling beams)": Iy_beam_pct,
    #"Ix_bC": Ix_beamC_pct,
    "Coupling Beam inertia in y": Iy_beamC_pct,
    #"J_bC": Jbc_pct,
    #"f11_s": f11s_pct,
    #"f22_s": f22s_pct,
    #"f12_s": f12s_pct,
    "Slab out-plane stiffness (m11)": m11s_pct,
    "Slab out-plane stiffness (m22)": m22s_pct,
    "Slab out-plane stiffness (m12)": m12s_pct,
    "Wall in-plane stiffness (f11)": [x + y for x, y in zip(f11w_pct,
f11w1_pct)],
    "Wall in-plane stiffness (f22)": [x + y for x, y in zip(f22w_pct,
f22w1_pct)],
    "Wall in-plane stiffness (f12)": [x + y for x, y in zip(f12w_pct,
f12w1_pct)]
    #"Beam torsional Constant\n(excl. coupling beams)": Jb_pct
}

# Construction de la matrice de données
param_labels = list(param_data.keys())
data_matrix = np.array([param_data[param][:n_modes] for param in
param_labels]).T  # shape: (14 modes x N paramètres)

# Palette personnalisée
colors = [
```

```
    (0.0, "#cce5ff"),  # bleu très pâle
    (0.3, "#66b3ff"),  # bleu moyen plus clair
    (1.0, "#0066cc")   # bleu foncé mais pas nuit
]
custom_cmap = LinearSegmentedColormap.from_list("custom_reds", colors, N=256)

# Création de la heatmap
plt.figure(figsize=(6, 15))
ax = sns.heatmap(
    data_matrix.T,  # <--- transpose ici
    cmap=custom_cmap,
    xticklabels=modes,          # ← maintenant sur l'axe X
    yticklabels=param_labels,   # ← maintenant sur l'axe Y
    cbar_kws={'label': 'Effet moyen μ* du paramètre sur la réponse modale
(%)'},
    annot=False,
    linewidths=0.5,
    linecolor='white'
)

# Récupérer la colorbar et modifier la taille du label et des ticks
cbar = ax.collections[0].colorbar
cbar.ax.tick_params(labelsize=15)


# Ajout manuel des annotations dans chaque cellule
for i in range(data_matrix.shape[1]):  # ← nombre de paramètres (lignes
maintenant)
    for j in range(data_matrix.shape[0]):  # ← nombre de modes (colonnes
maintenant)
        value = data_matrix[j, i]  # ← on accède à la valeur originale, non
transposée
        ax.text(j + 0.5, i + 0.5, f"{value:.2f}%",
                ha='center', va='center', color='black', fontsize=12)

# Tracer des lignes de séparation entre groupes de paramètres (axes inversés
= lignes)
# Les indices correspondent aux Y des lignes (entre les lignes, donc 1.5,
2.5, etc.)

# Liste des positions de séparation (lignes horizontales)
separators = [5, 9, 12, 15]

# Tracer les lignes
```

```
for sep in separators:
    # Ligne pleine dans la heatmap (zone des données uniquement)
    ax.hlines(sep, 0, len(modes), colors='black', linewidth=1.2)

    # Ligne pointillée juste à gauche de la 1ère colonne, sans changer les
axes
    ax.annotate(
        '',
        xy=(0, sep),
        xytext=(-2.7, sep),  # ← ajustable si tu veux plus long
        xycoords='data',
        textcoords='data',
        arrowprops=dict(
            arrowstyle='-',
            linestyle='dashed',
            color='black',
            linewidth=1.2
        )
    )

# Taille des ticks (étiquettes sur les axes X et Y)
ax.tick_params(axis='x', labelsize=12)
ax.tick_params(axis='y', labelsize=12)


plt.tight_layout()
plt.show()
plt.savefig("figure_for_word.png", dpi=300, bbox_inches='tight')


# ----------------------------------------------------------------------------
----------------------------------
# ZONE 5
# Charger le fichier Excel
file_path = r"z:/Cyrielle/4. Mémoire\Analyse de sensibilité/FINAL FINAL ZONE
5.xlsx"
xls = pd.ExcelFile(file_path)

# Créer un dictionnaire pour stocker les DataFrames par feuille
dfs_modes = {}

# Plage de lignes à extraire (A7 à A41)
start_row = 5
end_row = 40
```

```
param_col = 0
value_col = 3

# Lire chaque feuille et extraire les données souhaitées
for sheet in xls.sheet_names:
    df = pd.read_excel(xls, sheet_name=sheet)
    df_subset = df.iloc[start_row:end_row, [param_col, value_col]].copy()
    df_subset.columns = ["Paramètre", "Valeur"]
    df_subset.dropna(inplace=True)
    dfs_modes[sheet] = df_subset

# Exemple : afficher la feuille "freq_Freq_1"
print(dfs_modes["freq_Freq_1"])

df_mode1 = dfs_modes["freq_Freq_1"]
df_mode2 = dfs_modes["freq_Freq_2"]
df_mode3 = dfs_modes["freq_Freq_3"]
df_mode4 = dfs_modes["freq_Freq_4"]
df_mode5 = dfs_modes["freq_Freq_5"]
df_mode6 = dfs_modes["freq_Freq_6"]
df_mode7 = dfs_modes["freq_Freq_7"]
df_mode8 = dfs_modes["freq_Freq_8"]
df_mode9 = dfs_modes["freq_Freq_9"]
df_mode10 = dfs_modes["freq_Freq_10"]
df_mode11 = dfs_modes["freq_Freq_11"]
df_mode12 = dfs_modes["freq_Freq_12"]
df_mode13 = dfs_modes["freq_Freq_13"]
df_mode14 = dfs_modes["freq_Freq_14"]

# Convertir en tableaux numpy
ModuleYoungCol_pct = []
ModuleYoungCol1_pct = []
Ix_beam_pct = []
Iy_beam_pct = []
Ix_col_pct = []
Iy_col_pct = []
Ix_beamC_pct = []
Iy_beamC_pct = []
Jbc_pct = []
Densitebc_pct = []
Airebc_pct = []
Jb_pct = []
Densiteb_pct = []
Aireb_pct = []
```

```
Jc_pct = []
Densitec_pct = []
Airec_pct = []
ModuleYoungSlab_pct = []
Densites_pct = []
Thickness250_s_pct = []
Thickness225_s_pct = []
f11s_pct = []
f22s_pct = []
f12s_pct = []
m11s_pct = []
m22s_pct = []
m12s_pct = []
ModuleYoungWall_pct = []
ModuleYoungWall1_pct = []
ModuleYoungWall2_pct = []
Densitew_pct = []
f11w_pct = []
f22w_pct = []
f12w_pct = []
f11w1_pct = []
f22w1_pct = []
f12w1_pct = []

for i in range(5):
    ModuleYoungCol_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][5])*10
0)
    ModuleYoungCol1_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][6])*1
00)
    Ix_col_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][7])*100)
    Iy_col_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][8])*100)
    Jc_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][9])*100)
    ModuleYoungSlab_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][10])*
100)
    f11s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][11])*100)
    f22s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][12])*100)
    f12s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][13])*100)
    m11s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][14])*100)
    m22s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][15])*100)
    m12s_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][16])*100)
    ModuleYoungWall_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][17])*
100)
    ModuleYoungWall1_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][18])
*100)
```

```
    ModuleYoungWall2_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][19])
*100)
    f11w_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][20])*100)
    f22w_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][21])*100)
    f12w_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][22])*100)
    f11w1_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][23])*100)
    f22w1_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][24])*100)
    f12w1_pct.append((dfs_modes[f"freq_Freq_{i+1}"]["Valeur"][25])*100)

data_matrix = np.array([
    ModuleYoungCol_pct,
    ModuleYoungCol1_pct,
    ModuleYoungSlab_pct,
    ModuleYoungWall_pct,
    ModuleYoungWall1_pct,
    ModuleYoungWall2_pct,
    Ix_col_pct,
    Iy_col_pct,
    Jc_pct,
    f11s_pct, f22s_pct, f12s_pct,
    m11s_pct, m22s_pct, m12s_pct,
    f11w_pct, f22w_pct, f12w_pct,
    f11w1_pct, f22w1_pct, f12w1_pct
]).T  # shape: (14 modes x n paramètres)

# Données de base
n_modes = 5
modes = [f"Mode {i+1}" for i in range(n_modes)]

# Paramètres à inclure dans la heatmap (liste complète basée sur les noms des
listes existantes)
param_data = {
    "Column Young’s modulus": [a + b for a, b in zip(ModuleYoungCol_pct,
ModuleYoungCol1_pct)],
    "Slab Young’s modulus": ModuleYoungSlab_pct,
    "Wall Young’s modulus": [a + b + c for a, b, c in
zip(ModuleYoungWall_pct, ModuleYoungWall1_pct, ModuleYoungWall2_pct)],
    #"Ix_b": Ix_beam_pct,
    "Column inertia in x": Ix_col_pct,
    "Column inertia in y": Iy_col_pct,
    "Slab out-plane stiffness (m11)": m11s_pct,
    "Slab out-plane stiffness (m22)": m22s_pct,
    "Slab out-plane stiffness (m12)": m12s_pct,
    "Wall in-plane stiffness (f11)": f11w_pct+f11w1_pct,
```

```
    "Wall in-plane stiffness (f22)": f22w_pct+f22w1_pct,
    "Wall in-plane stiffness (f12)": f12w_pct+f12w1_pct,
    #"Beam torsional Constant\n(excl. coupling beams)": Jb_pct
}

# Construction de la matrice de données
param_labels = list(param_data.keys())
data_matrix = np.array([param_data[param][:n_modes] for param in
param_labels]).T  # shape: (14 modes x N paramètres)

# Palette personnalisée
# Palette mauve modérée (pas trop foncée à la fin)
colors = [
    (0.0, "#ecd6ff"),  # très pâle mauve
    (0.3, "#b266ff"),  # mauve moyen
    (1.0, "#5e2b97")   # mauve foncé bien saturé, mais pas noir
]
custom_cmap = LinearSegmentedColormap.from_list("custom_deep_purple", colors,
N=256)

# Création de la heatmap
plt.figure(figsize=(6, 15))
ax = sns.heatmap(
    data_matrix.T,  # <--- transpose ici
    cmap=custom_cmap,
    xticklabels=modes,          # ← maintenant sur l'axe X
    yticklabels=param_labels,   # ← maintenant sur l'axe Y
    cbar_kws={'label': 'Effet moyen μ* du paramètre sur la réponse modale
(%)'},
    annot=False,
    linewidths=0.5,
    linecolor='white'
)

# Récupérer la colorbar et modifier la taille du label et des ticks
cbar = ax.collections[0].colorbar
cbar.ax.tick_params(labelsize=15)


# Ajout manuel des annotations dans chaque cellule
for i in range(data_matrix.shape[1]):  # ← nombre de paramètres (lignes
maintenant)
    for j in range(data_matrix.shape[0]):  # ← nombre de modes (colonnes
maintenant)
```

```
        value = data_matrix[j, i]  # ← on accède à la valeur originale, non
transposée
        ax.text(j + 0.5, i + 0.5, f"{value:.2f}%",
                ha='center', va='center', color='black', fontsize=12)

# Tracer des lignes de séparation entre groupes de paramètres (axes inversés
= lignes)
# Les indices correspondent aux Y des lignes (entre les lignes, donc 1.5,
2.5, etc.)

# Liste des positions de séparation (lignes horizontales)
separators = [5, 9, 13]

# Tracer les lignes
for sep in separators:
    # Ligne pleine dans la heatmap (zone des données uniquement)
    ax.hlines(sep, 0, len(modes), colors='black', linewidth=1.2)

    # Ligne pointillée juste à gauche de la 1ère colonne, sans changer les
axes
    ax.annotate(
        '',
        xy=(0, sep),
        xytext=(-2.7, sep),  # ← ajustable si tu veux plus long
        xycoords='data',
        textcoords='data',
        arrowprops=dict(
            arrowstyle='-',
            linestyle='dashed',
            color='black',
            linewidth=1.2
        )
    )

# Taille des ticks (étiquettes sur les axes X et Y)
ax.tick_params(axis='x', labelsize=12)
ax.tick_params(axis='y', labelsize=12)


plt.tight_layout()
plt.show()
plt.savefig("figure_for_word.png", dpi=300, bbox_inches='tight')
```

## ANNEXE IV

## CALIBRATION : SCRIPT PYTHON

Ce code permet de réaliser la mise à jour du modèle numérique de la tour à l'aide de l'API d'ETABS, en ajustant les paramètres physiques jusqu'à convergence avec les résultats expérimentaux.

```
# -- coding utf-8 --

"""
01_run.py

File for performing FEM updating. Results from the FE model updating are
stored in the file info.pkl.
"""

import os
import sys
import pythonnet
from pathlib import Path
import comtypes.client
import shutil
import pickle
import time
import numpy as np
from SALib.sample.morris import sample
from SALib.analyze.morris import analyze
import matplotlib.pyplot as plt
import pytabs as pt
import pandas as pd
```

```
from scipy.optimize import lsq_linear
from collections import OrderedDict
from scipy.optimize import linear_sum_assignment


# --- Robust finite-diff & safety helpers ---
EPS_REL = 1e-3        # pas relatif pour les sensibilités
EPS_ABS = 1e-12       # plancher dénominateur

def safe_div(num, den, eps=EPS_ABS):
    """Divide with denominator floor to avoid inf/NaN."""
    den_abs = np.maximum(np.abs(den), eps)
    return num / den_abs

def all_finite(*arrs):
    return all(np.all(np.isfinite(a)) for a in arrs if a is not None)


# # ------------------------------------------------------------------------ #
# # MASS NORMALIZATION
# # ------------------------------------------------------------------------ #

def mass_normalize(phi, mass_vector):
    M = np.diag(mass_vector)
    normalized_phi = np.zeros_like(phi)
    for i in range(phi.shape[1]):  # pour chaque mode (colonne)
        factor = np.sqrt(phi[:, i].T @ M @ phi[:, i])
        normalized_phi[:, i] = phi[:, i] / factor
    return normalized_phi
```

```
file_path = r"Z:\Cyrielle\4. Mémoire\MODEL UPDATING\SEREP FINALE MODE 1.xlsx"

df_mass = pd.read_excel(file_path, sheet_name='Centers Of Mass And Rigidity')
Mass_vec_x = df_mass["Mass X"].to_numpy()
Mass_vec_y = df_mass["Mass Y"].to_numpy()
Mass_vec = np.concatenate((Mass_vec_x, Mass_vec_y), axis=0)

# ---------------------------------------------------------------------- #
# ANALYSIS SETTINGS
# ---------------------------------------------------------------------- #

# Number of iterations to perform
iterations = 10

# Measured frequencies to perform FE model updating on
mode_list = ['Mode 01', 'Mode 02', 'Mode 03', 'Mode 04', 'Mode 05'] # Five modes for the
model updating

# DEFINITIONS
def get_MAC_MMI(freqs_est, modes_est, freqs_num, modes_num, filtering=False,
                tol_upper=0.025, tol_lower=-0.025):
    """MAC and Mode Match Index (MMI) based on estimated and numerical results.

    This function establishes MAC and MMI and returns the information in
    matrices. The function also performs filtering of (local) numerical mode
    shapes based on a tolerance.

    Note that the format of the information contained in the estimated and
    numerical results for frequencies and mode shapes must be similar.
```

```
    Parameters
    ----------
    freqs_est : ndarray
        Estimated frequencies.
    modes_est : ndarray
        Estimated mode shapes.
    freqs_num : ndarray
        Numerically predicted frequencies.
    modes_num : ndarray
        Numerically predicted mode shapes.
    filtering : bool, optional
        When True, numerical mode shapes are filtered based on an upper and
        lower tolerance value. Default is False.
    tol_upper : float, optional
        Maximum upper tolerance value. Defaults to 0.025.
    tol_lower : float, optional
        Minimum lower tolerance value. Defaults to -0.025.

    Returns
    -------
    MAC_matrix : ndarray
        Filtered MAC matrix.
    MMI_matrix : ndarray
        Filtered MMI matrix.
    abaqus_filter_id_list : list
        List of id's of the filtered numerical mode shapes.
    results_MAC : dict
        Dictionary with the ten best MAC values (sorted) for each estimated
        mode shape evaluated.
```

```
results_MMI : dict
    Dictionary with the ten best MMI values (sorted) for each estimated
    mode shape evaluated.
"""

# ---------------- #
# MAC AND MMI
# ---------------- #

# General parameters
rows_est, cols_est = np.shape(modes_est)
rows_num, cols_num = np.shape(modes_num)

if filtering:

    # Abaqus parameters
    abaqus_filter_id_list = []
    tol_max = tol_upper
    tol_min = tol_lower

# MAC parameters
MAC_matrix = np.zeros((cols_num, cols_est), dtype=float)

# Mode match index parameters
gamma = 0.5
MMI_matrix = np.zeros((cols_num, cols_est), dtype=float)

# Loop through estimated mode shapes
for i in range(cols_est):
```

```
# Estimated mode shape vector and corresponding frequency considered
phi_e = modes_est[:, i]
f_e = freqs_est[i, 1]

# Loop through numerical mode shapes
for j in range(cols_num):

    if filtering:

        # Filter out local numerical modes
        mode_max = np.amax(modes_num[:, j])
        mode_min = np.amin(modes_num[:, j])

        # Check for local Abaqus mode shapes
        if mode_max < tol_max and mode_min > tol_min:

            # Append to list (only append one time, i == 0)
            if i == 0:
                abaqus_id = j+1
                abaqus_filter_id_list.append(abaqus_id)

            # Continue to next numerical mode shape
            continue

    # Numerical mode shaper vector and corresponding frequency
    # considered
    phi_n = modes_num[:, j]
    f_n = freqs_num[j, 1]

    # MAC value
```

```
        MAC_value = (np.abs(phi_n@phi_e))**2 / ((phi_n@phi_n)*(phi_e@phi_e))
        MAC = round(MAC_value, 5)

        # MMI value
        MMI = (1 - gamma)*MAC - gamma*(np.abs(f_e - f_n) / f_e)

        # Append
        MAC_matrix[j, i] = MAC
        MMI_matrix[j, i] = MMI

# ---------------- #
# SYSTEM ID
# ---------------- #

# Information matrix
SI = np.zeros((cols_num, 4), dtype=float)
SI[:, 2] = freqs_num[:, 1]
SI[:, 3] = np.arange(1, cols_num+1, 1)

# Dictionary with all results sorted by mode matching results and
# MAC results
results_MMI = OrderedDict([])
results_MAC = OrderedDict([])

# For each mode estimated by OMA, finding the maximum mode matching index
# (MMI) and the corresponding MAC, frequency and Abaqus ID.
for k in range(cols_est):

    # Estimated mode shape considered
    mode_est = 'Mode ' + str('{:02}'.format(k+1))
```

```
        # MMI
        SI[:, 0] = MMI_matrix[:, k]

        # MAC
        SI[:, 1] = MAC_matrix[:, k]

        # Sort by MMI
        idx_MMI = np.argsort(SI[:, 0], axis=0)
        SI_sort_MMI_asc = SI[idx_MMI]
        SI_sort_MMI_dsc = SI[idx_MMI][::-1]

        # Sort by MAC
        idx_MAC = np.argsort(SI[:, 1], axis=0)
        SI_sort_MAC_asc = SI[idx_MAC]
        SI_sort_MAC_dsc = SI[idx_MAC][::-1]

        # Append the 10 best results
        results_MMI.update({mode_est: SI_sort_MMI_dsc[:10]})
        results_MAC.update({mode_est: SI_sort_MAC_dsc[:10]})

    # ---------------- #
    # RETURN
    # ---------------- #

    # Return
    if filtering:

        return(MAC_matrix, MMI_matrix, abaqus_filter_id_list,
               results_MAC, results_MMI)
```

```
    else:

        return(MAC_matrix, MMI_matrix, results_MAC, results_MMI)


# Charger le fichier Excel
file_path = r"Z:\Cyrielle\4. Mémoire\FINAL FINAL MODEL USED FOR FINAL UPDATING\CALL\SEREP_RESULT_pyFBS.xlsx"
xls = pd.ExcelFile(file_path)

# Liste des noms de feuilles (une par mode)
sheet_names = xls.sheet_names[:-1]

# Dictionnaire pour stocker les DataFrames
mode_shapes = {}

# Boucle sur chaque feuille pour créer un DataFrame
for sheet in sheet_names:
    df = xls.parse(sheet)
    df.columns = ["U_exp_mode1 (SEREP + norm)", "U_num_mode1 (norm)"]
    mode_shapes[sheet] = df

# Initialiser des listes pour stocker les colonnes de chaque mode
U_exp_list = []
U_num_list = []

for sheet in sheet_names:
    df = mode_shapes[sheet]
    U_exp_list.append(df.iloc[:, 0].to_numpy())  # colonne expérimentale
```

```
    U_num_list.append(df.iloc[:, 1].to_numpy())  # colonne numérique

# Convertir en matrices numpy avec chaque colonne = un mode
U_exp_all_modes = np.column_stack(U_exp_list)
U_num_all_modes = np.column_stack(U_num_list)

# ---------------------------------------------------------------------------- #
# INITIAL
# ---------------------------------------------------------------------------- #

# Import measured frequencies and mode shapes
freqs_est = np.array([0.229, 0.250, 0.451, 0.925, 0.954])
modes_id = np.arange(1, 6)  # [1, 2, 3, 4, 5]
# Empiler les colonnes côte à côte (identifiant + fréquence)
freqs_est = np.column_stack((modes_id, freqs_est))

modes_est = U_exp_all_modes

# ---------------------------------- #
# FEM UPDATING PARAMETERS
# ---------------------------------- #

# # FEM initial updating parameters
# # Zone 5

# Ec1_5 = 38800 #MPa, 80Conccol
# Ec2_5 = 25000 #MPa, CONC
# Ec3_5 = 31000 #MPa, 45Conccol
# Ec4_5 = 25000 #MPa, CONC30
```

# Ew1_5 = 34600 #MPa, Fw_CONC60_5
# Ew2_5 = 26622.36 #MPa, Fw_CONC35_5

# Es1_5 = 25000 #MPa, CONC

m11S_5 = 0.35
m22S_5 = 0.35
m12S_5 = 0.35

f11W_5 = 0.95
f22W_5 = 0.95
f12W_5 = 0.95

# Zone 4
# Ec1_4 = 34600 #MPa, 60ConcCol_4
# Ec2_4 = 38800 #MPa, 80Conccol_4

# Ew1_4 = 34600 #MPa, CONC60_W_4

# Es1_4 = 25000 #MPa, CONC_S_4

m11S_4 = 0.35
m22S_4 = 0.35
m12S_4 = 0.35

f11W_4 = 0.95
f22W_4 = 0.95
f12W_4 = 0.95

# Zone 3

# Ec1_3 = 34600 #MPa, 60ConcCol_3

# Ew1_3 = 34600 #MPa, CONC60_W_3
# Ew2_3 = 31000 #MPa, CONC45_W_3

# Es1_3 = 25000 #MPa, CONC_S_2

m11S_3 = 0.35
m22S_3 = 0.35
m12S_3 = 0.35

f11W_3 = 0.95
f22W_3 = 0.95
f12W_3 = 0.95

# Zone 2
# Ec1_2 = 25000 #MPa, CONC_2
# Ec2_2 = 31000 #MPa, CONC45_2

# Ew1_2 = 31000 #MPa, CONC45_W_2

# Es1_2 = 25000 #MPa, CONC_S_2

m11S_2 = 0.35
m22S_2 = 0.35
m12S_2 = 0.35

f11W_2 = 0.95
f22W_2 = 0.95
f12W_2 = 0.95

```
# Zone 1
# Ec1_1 = 25000 #MPa, CONC30_1

# Ew1_1 = 25000 #MPa, CONC_W
# Ew2_1 = 31000 #MPa, CONC45_W_1

# Es1_1 = 25000 #MPa, CONC_S_1

m11S_1 = 0.35
m22S_1 = 0.35
m12S_1 = 0.35

f11W_1 = 0.95
f22W_1 = 0.95
f12W_1 = 0.95


# Parameter list


theta_0 = np.array([
    # Zone 1
    #Ec1_1, Ew1_1, Ew2_1, Es1_1,
    m11S_1, m22S_1, m12S_1, f11W_1, f22W_1, f12W_1,

    # Zone 2
    #Ec1_2, Ec2_2, Ew1_2, Es1_2,
    m11S_2, m22S_2, m12S_2, f11W_2, f22W_2, f12W_2,
```

```
    # Zone 3
    #Ec1_3, Ew1_3, Ew2_3, Es1_3,
    m11S_3, m22S_3, m12S_3, f11W_3, f22W_3, f12W_3,

    # Zone 4
    #Ec1_4, Ec2_4, Ew1_4, Es1_4,
    m11S_4, m22S_4, m12S_4, f11W_4, f22W_4, f12W_4,


    # Zone 5 – on ajoute ici plus de paramètres car plus de valeurs sont connues
    #Ec1_5, Ec2_5, Ec3_5, Ec4_5, Ew1_5, Ew2_5, Es1_5,
    m11S_5, m22S_5, m12S_5, f11W_5, f22W_5, f12W_5
])

# theta_0 = [31505.54475563,31503.71700948,37551.88222972,31538.17695778,
# 31503.71700935,37515.75416384,37515.75416369,31503.71700945,
# 41129.64768945,41121.10151375,37606.96170312,31511.70433359,
# 41122.02031917,45333.04939157,41163.40956716,31523.53481266,
# 45328.60532017,31505.1919007,37515.76011505,31521.27695527,
# 41253.13679951,33152.2305704,31503.71700949]


# theta_0 = np.array(theta_0)

# ---------------------------------- #
# FEM ANALYSIS
# ---------------------------------- #
```

```
#set the following flag to True to attach to an existing instance of the program
#otherwise a new instance of the program will be started
AttachToInstance = True

#this allows for a connection to a version of ETABS other than the latest installation
#otherwise the latest installed version of ETABS will be launched
SpecifyPath = True

#if the above flag is set to True, specify the path to ETABS below
ProgramPath = r"C:\Program Files\Computers and Structures\ETABS 22\ETABS.exe"

#full path to the model
#set it to the desired path of your model
APIPath = r"Z:\Cyrielle\4. Mémoire\FINAL FINAL MODEL USED FOR FINAL UPDATING\CALL\700SJ ELS stage construction final CALIBRATION1.EDB"
if not os.path.exists(APIPath):
    try:
        os.makedirs(APIPath)
    except OSError:
        pass
ModelPath = APIPath + os.sep + 'API_1-001.edb'

#create API helper object
helper = comtypes.client.CreateObject('ETABSv1.Helper')
helper = helper.QueryInterface(comtypes.gen.ETABSv1.cHelper)

if AttachToInstance:
    #attach to a running instance of ETABS
    try:
        #get the active ETABS object
```

```
        myETABSObject = helper.GetObject("CSI.ETABS.API.ETABSObject")
    except (OSError, comtypes.COMError):
        print("No running instance of the program found or failed to attach.")
        sys.exit(-1)
else:
    if SpecifyPath:
        try:
            #'create an instance of the ETABS object from the specified path
            myETABSObject = helper.CreateObject(ProgramPath)
        except (OSError, comtypes.COMError):
            print("Cannot start a new instance of the program from " + ProgramPath)
            sys.exit(-1)
    else:
        try:
            #create an instance of the ETABS object from the latest installed ETABS
            myETABSObject = helper.CreateObjectProgID("CSI.ETABS.API.ETABSObject")
        except (OSError, comtypes.COMError):
            print("Cannot start a new instance of the program.")
            sys.exit(-1)

    #start ETABS application
    myETABSObject.ApplicationStart()

#create SapModel object
SapModel = myETABSObject.SapModel

rep = SapModel

Param = theta_0
```

```
# Ec1_1 = Param[0] #MPa,
# Ew1_1 = Param[1] #MPa,
# Ew2_1 = Param[2] #MPa,
# Es1_1 = Param[3] #MPa,

m11S_1 = Param[0]
m22S_1 = Param[1]
m12S_1 = Param[2]
f11W_1 = Param[3]
f22W_1 = Param[4]
f12W_1 = Param[5]

# Ec1_2 = Param[4] # MPa
# Ec2_2 = Param[5]
# Ew1_2 = Param[6]
# Es1_2 = Param[7]

m11S_2 = Param[6]
m22S_2 = Param[7]
m12S_2 = Param[8]
f11W_2 = Param[9]
f22W_2 = Param[10]
f12W_2 = Param[11]

# Ec1_3 = Param[8] #MPa,
# Ew1_3 = Param[9] #MPa,
# Ew2_3 = Param[10] #MPa,
# Es1_3 = Param[11] #MPa,
```

```
m11S_3 = Param[12]
m22S_3 = Param[13]
m12S_3 = Param[14]
f11W_3 = Param[15]
f22W_3 = Param[16]
f12W_3 = Param[17]

# Ec1_4 = Param[12] #MPa,
# Ec2_4 = Param[13] #MPa,
# Ew1_4 = Param[14] #MPa,
# Es1_4 = Param[15] #MPa,

m11S_4 = Param[18]
m22S_4 = Param[19]
m12S_4 = Param[20]
f11W_4 = Param[21]
f22W_4 = Param[22]
f12W_4 = Param[23]

# Ec1_5 = Param[16] #MPa,
# Ec2_5 = Param[17] #MPa,
# Ec3_5 = Param[18] #MPa,
# Ec4_5 = Param[19] #MPa,
# Ew1_5 = Param[20] #MPa,
# Ew2_5 = Param[21] #MPa,
# Es1_5 = Param[22] #MPa,

m11S_5 = Param[24]
m22S_5 = Param[25]
m12S_5 = Param[26]
```

```
f11W_5 = Param[27]
f22W_5 = Param[28]
f12W_5 = Param[29]


# ret = SapModel.PropMaterial.SetMPIsotropic('80Conccol_5', Ec1_5*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC_5', Ec2_5*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('45ConcCol_5', Ec3_5*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC30_5', Ec4_5*1000, 0.2, 0.0000099)

# ret = SapModel.PropMaterial.SetMPIsotropic('FW_CONC60_5', Ew1_5*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('FW_CONC35_5', Ew2_5*1000, 0.2, 0.0000099)

# ret = SapModel.PropMaterial.SetMPIsotropic('CONC_S_5', Es1_5*1000, 0.2, 0.0000099)


# PENSER à METTRE TOUS LES MODIFIERS DE LA ZONE ÉTUDIÉE AU DEBUT
# + S'ASSURER QUE CEUX DES AUTRES ZONES ONT LES BONS
Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DALLE200_5", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DALLE225_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DALLE250_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DALLE300_5", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DALLE325_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DALLE350_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DALLE400_5", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DALLE450_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DROP250_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DROP350_5", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DROP450_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DROP475_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DROP50_5", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("FW1", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("FW2", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("FW3", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW1", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW10", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW2", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW3", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW4", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW5", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW6", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW7", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW8", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW9", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("W1", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("W2", Value.tolist())
```

```
# ret = SapModel.PropMaterial.SetMPIsotropic('60ConcCol_4', Ec1_4*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('80Conccol_4', Ec2_4*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC60_W_4', Ew1_4*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC_S_4', Es1_4*1000, 0.2, 0.0000099)

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_4
Value[4] = m22S_4
Value[5] = m12S_4
ret = SapModel.PropArea.SetModifiers("DALLE250_4", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_4
Value[4] = m22S_4
Value[5] = m12S_4
ret = SapModel.PropArea.SetModifiers("DROP450_4", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_4
```

```
Value[1] = f22W_4
Value[2] = f12W_4
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW19_4", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_4
Value[1] = f22W_4
Value[2] = f12W_4
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW18_4", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_4
Value[1] = f22W_4
Value[2] = f12W_4
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW17_4", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_4
```

```
Value[1] = f22W_4
Value[2] = f12W_4
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW20_4", Value.tolist())

# ret = SapModel.PropMaterial.SetMPIsotropic('60ConcCol_3', Ec1_3*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC60_W_3', Ew1_3*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC45_W_3', Ew2_3*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC_S_3', Es1_3*1000, 0.2, 0.0000099)


Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_3
Value[4] = m22S_3
Value[5] = m12S_3
ret = SapModel.PropArea.SetModifiers("DALLE250_3", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
```

```
Value[3] = m11S_3
Value[4] = m22S_3
Value[5] = m12S_3
ret = SapModel.PropArea.SetModifiers("DROP450_3", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_3
Value[1] = f22W_3
Value[2] = f12W_3
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW16_3", Value.tolist())

# ret = SapModel.PropMaterial.SetMPIsotropic('CONC_2', Ec1_2*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC45_2', Ec2_2*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC45_W_2', Ew1_2*1000, 0.2,
0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC_S_2', Es1_2*1000, 0.2, 0.0000099)

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_2
Value[4] = m22S_2
Value[5] = m12S_2
ret = SapModel.PropArea.SetModifiers("DALLE250_2", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_2
Value[4] = m22S_2
Value[5] = m12S_2
ret = SapModel.PropArea.SetModifiers("DROP450_2", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_2
Value[1] = f22W_2
Value[2] = f12W_2
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW15_2", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_2
Value[1] = f22W_2
Value[2] = f12W_2
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW13_2", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = f11W_2
Value[1] = f22W_2
Value[2] = f12W_2
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW16_2", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_2
Value[1] = f22W_2
Value[2] = f12W_2
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW14_2", Value.tolist())


# ret = SapModel.PropMaterial.SetMPIsotropic('CONC30_1', Ec1_1*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC_W', Ew1_1*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC45_W_1', Ew2_1*1000, 0.2,
0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC_S_1', Es1_1*1000, 0.2, 0.0000099)

Value = np.ones((10))
```

```
Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_1
Value[4] = m22S_1
Value[5] = m12S_1
ret = SapModel.PropArea.SetModifiers("DALLE250_1", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_1
Value[4] = m22S_1
Value[5] = m12S_1
ret = SapModel.PropArea.SetModifiers("DALLE225_1", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_1
Value[4] = m22S_1
Value[5] = m12S_1
ret = SapModel.PropArea.SetModifiers("DROP450_1", Value.tolist())

Value = np.ones((10))
```

```
Value[0] = f11W_1
Value[1] = f22W_1
Value[2] = f12W_1
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW22_1", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_1
Value[1] = f22W_1
Value[2] = f12W_1
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW21_1", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_1
Value[1] = f22W_1
Value[2] = f12W_1
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW15_1", Value.tolist())

Value = np.ones((10))
```

```
Value[0] = f11W_1
Value[1] = f22W_1
Value[2] = f12W_1
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW13_1", Value.tolist())


ret = SapModel.File.Save('Z:\\Cyrielle\\4. Mémoire\\FINAL FINAL MODEL USED FOR FINAL UPDATING\\CALL\\700SJ ELS stage construction final CALIBRATION1.EDB')

ModalTable = []
ModeShape = []
ModalTableFINAL = []
ModeShapeFINAL = []

#Exécuter l'analyse modale
SapModel.Analyze.RunAnalysis()

ret = SapModel.Results.Setup.SetCaseSelectedForOutput("MODAL")
ret = SapModel.Results.Setup.SetOptionModeShape(1, 15, True)

NumberResults = 0
Obj = ""
Elm = ""
LoadCase = "MODAL"
StepType = ""
StepNum = []
U1 = []
```

```
U2 = []
U3 = []
R1 = []
R2 = []
R3 = []

TABLE = SapModel.Results.JointDispl("ALL", 2, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
TABLE = TABLE[5:8]
x = np.array(TABLE[0])
y1 = np.array(TABLE[1])
y2 = np.array(TABLE[2])

valeurs_uniques = np.unique(x)

# Résultat : liste de tuples (valeur x, max abs y1, max abs y2)
resultats = []

for val in valeurs_uniques:
    indices = np.where(x == val)[0]
    max_y1 = np.max(np.abs(y1[indices]))
    max_y2 = np.max(np.abs(y2[indices]))
    resultats.append((val, max_y1, max_y2))

ret = SapModel.Results.JointDispl("7442", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7444", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
```

```
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7445", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7446", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7447", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7448", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7451", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7449", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7450", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)
```

```
ret = SapModel.Results.JointDispl("7454", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7455", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7458", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7459", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7461", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7462", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7463", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7464", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
```

```
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7465", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7466", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7467", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7471", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7472", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7473", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7474", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)
```

```
ret = SapModel.Results.JointDispl("7475", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7476", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7477", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7478", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7479", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7480", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7481", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7482", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
```

```
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7483", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7484", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7485", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7486", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7487", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7488", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7489", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)
```

```
ret = SapModel.Results.JointDispl("7490", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7491", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7492", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7493", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7494", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7495", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7496", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7497", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
```

```
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7531", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7499", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7500", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7501", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7502", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7503", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7504", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)
```

```
ret = SapModel.Results.JointDispl("7505", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7506", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7507", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7508", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7509", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7510", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7511", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7513", 1, NumberResults, Obj, Elm, LoadCase, StepType,
StepNum, U1, U2, U3, R1, R2, R3)
```

```
ModeShape.append(ret)


ModeShapeFINAL.append(ModeShape.copy())
ModeShapeITE = ModeShapeFINAL[0]


Ux = []
for i in range(len(ModeShapeITE)):
    Ux.append(ModeShapeITE[i][6])

Ux = np.array(Ux)

for i in range(20):
    Ux[:,i] = Ux[:,i]/resultats[i][1]

Uy = []
for i in range(len(ModeShapeITE)):
    Uy.append(ModeShapeITE[i][7])

Uy = np.array(Uy)

for i in range(20):
    Uy[:,i] = Uy[:,i]/resultats[i][2]

modes_num = np.vstack((Ux, Uy))
modes_num = modes_num[:, :5]
modes_num = mass_normalize(modes_num,Mass_vec)

ret = SapModel.Results.Setup.SetCaseSelectedForOutput("MODAL")
```

```
# Initialiser les variables
NumberResults = 0  # Cette variable sera remplie par la méthode ModalPeriod()

# Les arrays doivent être de type spécifique. Ici, nous utilisons array de type 'd' pour les doubles
et 'u' pour les strings
LoadCase = ""
StepType = ""
StepNum = []
Period = []
Frequency = []
CircFreq = []
EigenValue = []

ret = SapModel.Results.ModalPeriod(NumberResults, LoadCase, StepType, StepNum,
Period, Frequency, CircFreq, EigenValue)
ModalTable.append(ret)
ModalTableFINAL.append(ModalTable.copy())

for i in range(5):
    Frequency.append(ModalTableFINAL[0][0][5][i])

# Convertir en array colonne
frequency_array = np.array(Frequency).reshape(-1, 1)
mode_ids = np.arange(1, len(Frequency) + 1).reshape(-1, 1)
freqs_num = np.hstack((mode_ids, frequency_array))

ret = SapModel.SetModelIsLocked(False)

(MAC_initial_model, _, results_MAC, _) = get_MAC_MMI(freqs_est, modes_est,
```

```
                                    freqs_num, modes_num,
                                    filtering=False)

# General parameters
rows_est, cols_est = np.shape(modes_est)
rows_num, cols_num = np.shape(modes_num)

# ----------------------------------------------------------------------#
# FEM UPDATING
# ----------------------------------------------------------------------#

# ---------------------------------- #
# INITIALIZATION
# ---------------------------------- #

# Variables
q = len(mode_list)
p = len(theta_0)

# Declaration of variables
# Variables saved for each iteration
theta = np.zeros((iterations + 1, p), dtype=float)
lamda = np.zeros((iterations + 1, q), dtype=float)
lamda_all = np.zeros((iterations + 1, len(freqs_est)), dtype=float)
lamda_m = np.zeros(len(mode_list), dtype=float)

G_matrices = OrderedDict([])
MAC_updated_model = OrderedDict([])
results_MAC_updated_model = OrderedDict([])
results_MAC_pt_all_matrices = OrderedDict([])
```

```
J_star_list = []

# Start values for the iterations
for idx, mode in enumerate(mode_list):

    # Establish frequencies based on the mode_list string
    string_idx = mode[-2:]

    # Correct for 0 in the string
    if string_idx[0] == 0:

        f_idx = int(mode[-1]) - 1

    else:
        f_idx = int(string_idx) - 1

    # Measured frequencies (to perform FEM updating on)
    lamda_m[idx] = freqs_est[:, 1][f_idx]

    # Numerical frequencies (to perform FEM updating on)
    # Top match based on MAC
    lamda[0, idx] = results_MAC[mode][0, 2]

# All numerical frequencies
for idx, mode in enumerate(results_MAC.keys()):

    lamda_all[0, idx] = results_MAC[mode][0, 2]

# Updating parameters
theta_0 = np.array(theta_0, dtype=float)
```

```
theta[0, :] = np.copy(theta_0)
d_theta_start = 0.01 * theta_0

# Weighting matrix
W = np.diag(1.0 / np.maximum(lamda_m, 1e-12)**2)



# ------------------------------------ #
# OBJECTIVE FUNCTION
# ------------------------------------ #

# Poids
wf = 1
wmac = 1

# MAC : diagonale uniquement
MAC_diag = np.diag(MAC_initial_model)
J_mac = np.sum((1 - MAC_diag)**2)

# Evaluate the objective function for the initial analysis
J_star = np.sum(np.diag(W)*((lamda_m - lamda[0, :])/lamda[0, :])**2) * wf + wmac * J_mac
J_star_list.append(J_star)

# ------------------------------------ #
# ITERATIONS
# ------------------------------------ #

for itr in range(10): #iterations

    # First iteration step
```

```
    if itr == 0:

        # Initialize the parameter incremenet
        d_theta = d_theta_start

    # All other iteration steps
    else:

        # Update the parameter increment
        d_theta = d_theta_upd

    # Variable
    results_MAC_pt_matrices = OrderedDict([])

    # ------------------------- #
    # SENSITIVITY MATRIX
    # ------------------------- #

    # Sensitivity matrix (scaled)
    G = np.zeros((q, p), dtype=float)

    # Loop through the parameters (columns)
    for ii in range(p):

        # ------------------------- #
        # PERTURBATION - THETA
        # ------------------------- #

        # Perturbed parameter vector
        theta_pt = np.copy(theta[itr, :])
```

```
# pas absolu = eps_rel * max(|theta_cur|, |theta0|, 1e-8)
base = max(abs(theta[itr, ii]), abs(theta_0[ii]), 1e-8)
delta = EPS_REL * base

# oriente le pas (facultatif): même signe que la valeur courante
theta_pt[ii] = theta_pt[ii] + np.copysign(delta, theta[itr, ii] if theta[itr, ii] != 0 else 1.0)

# ------------------------- #
# PERTURBATION - LAMDA
# ------------------------- #

# ------------------------- #
# FE ANALYSIS
# ------------------------- #

#set the following flag to True to attach to an existing instance of the program
#otherwise a new instance of the program will be started
AttachToInstance = True

#this allows for a connection to a version of ETABS other than the latest installation
#otherwise the latest installed version of ETABS will be launched
SpecifyPath = True

#if the above flag is set to True, specify the path to ETABS below
ProgramPath = r"C:\Program Files\Computers and Structures\ETABS 22\ETABS.exe"

#full path to the model
#set it to the desired path of your model
```

```
    APIPath = r"Z:\Cyrielle\4. Mémoire\FINAL FINAL MODEL USED FOR FINAL UPDATING\700SJ ELS stage construction final CALIBRATION"
    if not os.path.exists(APIPath):
        try:
            os.makedirs(APIPath)
        except OSError:
            pass
    ModelPath = APIPath + os.sep + 'API_1-001.edb'

    #create API helper object
    helper = comtypes.client.CreateObject('ETABSv1.Helper')
    helper = helper.QueryInterface(comtypes.gen.ETABSv1.cHelper)

    if AttachToInstance:
        #attach to a running instance of ETABS
        try:
            #get the active ETABS object
            myETABSObject = helper.GetObject("CSI.ETABS.API.ETABSObject")
        except (OSError, comtypes.COMError):
            print("No running instance of the program found or failed to attach.")
            sys.exit(-1)
    else:
        if SpecifyPath:
            try:
                #'create an instance of the ETABS object from the specified path
                myETABSObject = helper.CreateObject(ProgramPath)
            except (OSError, comtypes.COMError):
                print("Cannot start a new instance of the program from " + ProgramPath)
                sys.exit(-1)
        else:
```

```
            try:
                #create an instance of the ETABS object from the latest installed ETABS
                myETABSObject                                                    =
helper.CreateObjectProgID("CSI.ETABS.API.ETABSObject")
            except (OSError, comtypes.COMError):
                print("Cannot start a new instance of the program.")
                sys.exit(-1)

            #start ETABS application
            myETABSObject.ApplicationStart()

        #create SapModel object
        SapModel = myETABSObject.SapModel

        rep = SapModel

        Param = theta_pt
        # Ec1_1 = Param[0] #MPa,
        # Ew1_1 = Param[1] #MPa,
        # Ew2_1 = Param[2] #MPa,
        # Es1_1 = Param[3] #MPa,

        m11S_1 = Param[0]
        m22S_1 = Param[1]
        m12S_1 = Param[2]
        f11W_1 = Param[3]
        f22W_1 = Param[4]
        f12W_1 = Param[5]

        # Ec1_2 = Param[4] # MPa
```

```
# Ec2_2 = Param[5]
# Ew1_2 = Param[6]
# Es1_2 = Param[7]

m11S_2 = Param[6]
m22S_2 = Param[7]
m12S_2 = Param[8]
f11W_2 = Param[9]
f22W_2 = Param[10]
f12W_2 = Param[11]

# Ec1_3 = Param[8] #MPa,
# Ew1_3 = Param[9] #MPa,
# Ew2_3 = Param[10] #MPa,
# Es1_3 = Param[11] #MPa,

m11S_3 = Param[12]
m22S_3 = Param[13]
m12S_3 = Param[14]
f11W_3 = Param[15]
f22W_3 = Param[16]
f12W_3 = Param[17]

# Ec1_4 = Param[12] #MPa,
# Ec2_4 = Param[13] #MPa,
# Ew1_4 = Param[14] #MPa,
# Es1_4 = Param[15] #MPa,

m11S_4 = Param[18]
m22S_4 = Param[19]
```

```
m12S_4 = Param[20]
f11W_4 = Param[21]
f22W_4 = Param[22]
f12W_4 = Param[23]

# Ec1_5 = Param[16] #MPa,
# Ec2_5 = Param[17] #MPa,
# Ec3_5 = Param[18] #MPa,
# Ec4_5 = Param[19] #MPa,
# Ew1_5 = Param[20] #MPa,
# Ew2_5 = Param[21] #MPa,
# Es1_5 = Param[2] #MPa,

m11S_5 = Param[24]
m22S_5 = Param[25]
m12S_5 = Param[26]
f11W_5 = Param[27]
f22W_5 = Param[28]
f12W_5 = Param[29]



# ret = SapModel.PropMaterial.SetMPIsotropic('80Conccol_5', Ec1_5*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC_5', Ec2_5*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('45ConcCol_5', Ec3_5*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC30_5', Ec4_5*1000, 0.2, 0.0000099)
```

```
    # ret = SapModel.PropMaterial.SetMPIsotropic('FW_CONC60_5', Ew1_5*1000, 0.2, 0.0000099)
    # ret = SapModel.PropMaterial.SetMPIsotropic('FW_CONC35_5', Ew2_5*1000, 0.2, 0.0000099)

    # ret = SapModel.PropMaterial.SetMPIsotropic('CONC_S_5', Es1_5*1000, 0.2, 0.0000099)


    # PENSER à METTRE TOUS LES MODIFIERS DE LA ZONE ÉTUDIÉE AU DEBUT
    # + S'ASSURER QUE CEUX DES AUTRES ZONES ONT LES BONS
    Value = np.ones((10))

    Value[0] = 0.35
    Value[1] = 0.35
    Value[2] = 0.35
    Value[3] = m11S_5
    Value[4] = m22S_5
    Value[5] = m12S_5
    ret = SapModel.PropArea.SetModifiers("DALLE200_5", Value.tolist())

    Value = np.ones((10))

    Value[0] = 0.35
    Value[1] = 0.35
    Value[2] = 0.35
    Value[3] = m11S_5
    Value[4] = m22S_5
    Value[5] = m12S_5
```

```
ret = SapModel.PropArea.SetModifiers("DALLE225_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DALLE250_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DALLE300_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
```

```
ret = SapModel.PropArea.SetModifiers("DALLE325_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DALLE350_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DALLE400_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
```

```
ret = SapModel.PropArea.SetModifiers("DALLE450_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DROP250_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DROP350_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
```

```
ret = SapModel.PropArea.SetModifiers("DROP450_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DROP475_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DROP50_5", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
```

```
ret = SapModel.PropArea.SetModifiers("FW1", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("FW2", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("FW3", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
```

```
ret = SapModel.PropArea.SetModifiers("SW1", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW10", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW2", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
```

```
ret = SapModel.PropArea.SetModifiers("SW3", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW4", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW5", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
```

```
ret = SapModel.PropArea.SetModifiers("SW6", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW7", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW8", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
```

```
ret = SapModel.PropArea.SetModifiers("SW9", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("W1", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("W2", Value.tolist())


# ret = SapModel.PropMaterial.SetMPIsotropic('60ConcCol_4', Ec1_4*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('80Conccol_4', Ec2_4*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC60_W_4', Ew1_4*1000, 0.2, 0.0000099)
```

```
    # ret = SapModel.PropMaterial.SetMPIsotropic('CONC_S_4', Es1_4*1000, 0.2, 0.0000099)

    Value = np.ones((10))

    Value[0] = 0.35
    Value[1] = 0.35
    Value[2] = 0.35
    Value[3] = m11S_4
    Value[4] = m22S_4
    Value[5] = m12S_4
    ret = SapModel.PropArea.SetModifiers("DALLE250_4", Value.tolist())

    Value = np.ones((10))

    Value[0] = 0.35
    Value[1] = 0.35
    Value[2] = 0.35
    Value[3] = m11S_4
    Value[4] = m22S_4
    Value[5] = m12S_4
    ret = SapModel.PropArea.SetModifiers("DROP450_4", Value.tolist())

    Value = np.ones((10))

    Value[0] = f11W_4
    Value[1] = f22W_4
    Value[2] = f12W_4
    Value[3] = 0.1
    Value[4] = 0.1
```

```
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW19_4", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_4
Value[1] = f22W_4
Value[2] = f12W_4
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW18_4", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_4
Value[1] = f22W_4
Value[2] = f12W_4
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW17_4", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_4
Value[1] = f22W_4
Value[2] = f12W_4
Value[3] = 0.1
Value[4] = 0.1
```

```
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW20_4", Value.tolist())

# ret = SapModel.PropMaterial.SetMPIsotropic('60ConcCol_3', Ec1_3*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC60_W_3', Ew1_3*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC45_W_3', Ew2_3*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC_S_3', Es1_3*1000, 0.2, 0.0000099)


Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_3
Value[4] = m22S_3
Value[5] = m12S_3
ret = SapModel.PropArea.SetModifiers("DALLE250_3", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_3
Value[4] = m22S_3
```

```
Value[5] = m12S_3
ret = SapModel.PropArea.SetModifiers("DROP450_3", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_3
Value[1] = f22W_3
Value[2] = f12W_3
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW16_3", Value.tolist())

# ret = SapModel.PropMaterial.SetMPIsotropic('CONC_2', Ec1_2*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC45_2', Ec2_2*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC45_W_2', Ew1_2*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC_S_2', Es1_2*1000, 0.2, 0.0000099)

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_2
Value[4] = m22S_2
Value[5] = m12S_2
```

```
ret = SapModel.PropArea.SetModifiers("DALLE250_2", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_2
Value[4] = m22S_2
Value[5] = m12S_2
ret = SapModel.PropArea.SetModifiers("DROP450_2", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_2
Value[1] = f22W_2
Value[2] = f12W_2
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW15_2", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_2
Value[1] = f22W_2
Value[2] = f12W_2
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
```

```
ret = SapModel.PropArea.SetModifiers("SW13_2", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_2
Value[1] = f22W_2
Value[2] = f12W_2
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW16_2", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_2
Value[1] = f22W_2
Value[2] = f12W_2
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW14_2", Value.tolist())


# ret = SapModel.PropMaterial.SetMPIsotropic('CONC30_1', Ec1_1*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC_W', Ew1_1*1000, 0.2, 0.0000099)
# ret = SapModel.PropMaterial.SetMPIsotropic('CONC45_W_1', Ew2_1*1000, 0.2, 0.0000099)
```

```
    # ret = SapModel.PropMaterial.SetMPIsotropic('CONC_S_1', Es1_1*1000, 0.2, 0.0000099)

    Value = np.ones((10))

    Value[0] = 0.35
    Value[1] = 0.35
    Value[2] = 0.35
    Value[3] = m11S_1
    Value[4] = m22S_1
    Value[5] = m12S_1
    ret = SapModel.PropArea.SetModifiers("DALLE250_1", Value.tolist())

    Value = np.ones((10))

    Value[0] = 0.35
    Value[1] = 0.35
    Value[2] = 0.35
    Value[3] = m11S_1
    Value[4] = m22S_1
    Value[5] = m12S_1
    ret = SapModel.PropArea.SetModifiers("DALLE225_1", Value.tolist())

    Value = np.ones((10))

    Value[0] = 0.35
    Value[1] = 0.35
    Value[2] = 0.35
    Value[3] = m11S_1
    Value[4] = m22S_1
```

```
Value[5] = m12S_1
ret = SapModel.PropArea.SetModifiers("DROP450_1", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_1
Value[1] = f22W_1
Value[2] = f12W_1
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW22_1", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_1
Value[1] = f22W_1
Value[2] = f12W_1
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW21_1", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_1
Value[1] = f22W_1
Value[2] = f12W_1
Value[3] = 0.1
Value[4] = 0.1
```

```
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW15_1", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_1
Value[1] = f22W_1
Value[2] = f12W_1
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW13_1", Value.tolist())

ret = SapModel.File.Save('Z:\\Cyrielle\\4. Mémoire\\FINAL FINAL MODEL USED FOR FINAL UPDATING\\CALL\\700SJ ELS stage construction final CALIBRATION1.EDB')


# ---------------- #
# EXPORT VARIABLES
# ---------------- #

# Job name
job_name = 'Analysis' + str(itr+1) + '_pt' + str(ii+1)
with open('job_name_list' + '.txt', 'a') as fout:
    fout.write('\n' + job_name)

# Parameter list
np.save('parameter_list_pt' + '.npy', theta_pt, allow_pickle=True,
        fix_imports=True)
```

```
# ---------------- #
# RUN ANALYSIS
# ---------------- #

# ------------------------- #
# POST-PROCESS
# ------------------------- #

ModalTable = []
ModeShape = []
ModalTableFINAL = []
ModeShapeFINAL = []

#Exécuter l'analyse modale
SapModel.Analyze.RunAnalysis()

ret = SapModel.Results.Setup.SetCaseSelectedForOutput("MODAL")
ret = SapModel.Results.Setup.SetOptionModeShape(1, 15, True)

NumberResults = 0
Obj = ""
Elm = ""
LoadCase = "MODAL"
StepType = ""
StepNum = []
U1 = []
U2 = []
U3 = []
R1 = []
```

```
    R2 = []
    R3 = []

    TABLE = SapModel.Results.JointDispl("ALL", 2, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    TABLE = TABLE[5:8]
    x = np.array(TABLE[0])
    y1 = np.array(TABLE[1])
    y2 = np.array(TABLE[2])

    valeurs_uniques = np.unique(x)

    # Résultat : liste de tuples (valeur x, max abs y1, max abs y2)
    resultats = []

    for val in valeurs_uniques:
        indices = np.where(x == val)[0]
        max_y1 = np.max(np.abs(y1[indices]))
        max_y2 = np.max(np.abs(y2[indices]))
        resultats.append((val, max_y1, max_y2))

    ret = SapModel.Results.JointDispl("7442", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7444", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)
```

```
ret = SapModel.Results.JointDispl("7445", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7446", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7447", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7448", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7451", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7449", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7450", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7454", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
```

```
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7455", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7458", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7459", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7461", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7462", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7463", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7464", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)
```

```
ret = SapModel.Results.JointDispl("7465", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7466", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7467", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7471", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7472", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7473", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7474", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7475", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
```

```
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7476", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7477", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7478", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7479", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7480", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7481", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7482", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)
```

```
ret = SapModel.Results.JointDispl("7483", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7484", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7485", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7486", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7487", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7488", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7489", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7490", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
```

```
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7491", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7492", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7493", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7494", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7495", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7496", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7497", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)
```

```
ret = SapModel.Results.JointDispl("7531", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7499", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7500", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7501", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7502", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7503", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7504", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
ModeShape.append(ret)

ret = SapModel.Results.JointDispl("7505", 1, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
```

```
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7506", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7507", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7508", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7509", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7510", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7511", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7513", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)
```

```
ModeShapeFINAL.append(ModeShape.copy())
ModeShapeITE = ModeShapeFINAL[0]


Ux = []
for i in range(len(ModeShapeITE)):
    Ux.append(ModeShapeITE[i][6])

Ux = np.array(Ux)

for i in range(20):
    Ux[:,i] = Ux[:,i]/resultats[i][1]

Uy = []
for i in range(len(ModeShapeITE)):
    Uy.append(ModeShapeITE[i][7])

Uy = np.array(Uy)

for i in range(20):
    Uy[:,i] = Uy[:,i]/resultats[i][2]

modes_num_pt = np.vstack((Ux, Uy))
modes_num_pt = modes_num_pt[:, :5]
modes_num_pt = mass_normalize(modes_num_pt,Mass_vec)

ret = SapModel.Results.Setup.SetCaseSelectedForOutput("MODAL")

# Initialiser les variables
```

```
    NumberResults = 0  # Cette variable sera remplie par la méthode ModalPeriod()

    # Les arrays doivent être de type spécifique. Ici, nous utilisons array de type 'd' pour les doubles et 'u' pour les strings
    LoadCase = ""
    StepType = ""
    StepNum = []
    Period = []
    Frequency = []
    CircFreq = []
    EigenValue = []

    ret = SapModel.Results.ModalPeriod(NumberResults, LoadCase, StepType, StepNum, Period, Frequency, CircFreq, EigenValue)
    ModalTable.append(ret)
    ModalTableFINAL.append(ModalTable.copy())

    for i in range(5):
       Frequency.append(ModalTableFINAL[0][0][5][i])

    # Convertir en array colonne
    frequency_array = np.array(Frequency).reshape(-1, 1)
    mode_ids = np.arange(1, len(Frequency) + 1).reshape(-1, 1)
    freqs_num_pt = np.hstack((mode_ids, frequency_array))

    ret = SapModel.SetModelIsLocked(False)

    # Sorted MAC and MMI
    (_, _, results_MAC_pt, _) = get_MAC_MMI(freqs_est, modes_est,
                                            freqs_num_pt, modes_num_pt,
```

```
                                filtering=False)

# Append perturbed results for validation
results_MAC_pt_matrices.update({'results_MAC_pt' + str(ii+1) :
                                results_MAC_pt})

# Perturbed numerical frequencies
lamda_pt = np.zeros(q, dtype=float)
lamda_pt_all = np.zeros(cols_est, dtype=float)

for idx, mode in enumerate(mode_list):

    # Establish frequencies based on the mode_list string
    string_idx = mode[-2:]

    # Correct for 0 in the string
    if string_idx[0] == 0:

        f_idx = int(mode[-1]) - 1

    else:
        f_idx = int(string_idx) - 1

    # Perturbed numerical frequencies
    # Top match based on MAC
    lamda_pt[idx] = results_MAC_pt[mode][0, 2]

# All perturbed numerical frequencies
for idx, mode in enumerate(results_MAC_pt.keys()):
```

```
        lamda_pt_all[idx] = results_MAC_pt[mode][0, 2]

    # Sensitivities
    # Loop through the frequencies (rows)
    for jj in range(q):

        nmr = (lamda_pt[jj] - lamda[itr, jj]) / max(abs(lamda[itr, jj]), 1e-12)
        dnm_raw = (theta_pt[ii] - theta[itr, ii])
        dnm = np.sign(dnm_raw) * max(abs(dnm_raw), EPS_ABS)   # clamp

        # Update sensitivity matrix
        G[jj, ii] = nmr/dnm
        # Nettoyage: remplace NaN/Inf par 0 pour éviter les plantages du solveur
        bad = ~np.isfinite(G)
        if np.any(bad):
            G[bad] = 0.0

# Append
G_matrices.update({'G' + str(itr+1): G})

# Append
results_MAC_pt_all_matrices.update({'results_MAC_pt_matrices_' +
                                    str(itr+1): results_MAC_pt_matrices})



# ------------------------- #
# OPTIMIZATION
# ------------------------- #

# 1) Unique mode pairing (Hungarian) using MMI
```

```
    MAC_mat, MMI_mat, _, _ = get_MAC_MMI(freqs_est, modes_est,
                                freqs_num_pt, modes_num_pt,
                                filtering=False)
    ri, ci = linear_sum_assignment(-MMI_mat)
    # Réordonner:
    freqs_num_assigned = freqs_num_pt[ri, :]          # colonnes déjà ok
    MAC_assigned = MAC_mat[ri][:, ci]
    lamda_cur = freqs_num_assigned[:, 1]                 # ← fréquence num pour chaque mode
estimé
    MAC_diag_assigned = np.diag(MAC_assigned)

    # 2) Off-diagonal penalty (limits mode collisions)
    off = MAC_assigned.copy()
    np.fill_diagonal(off, 0.0)

    # 3) Per-mode weights (boost modes 3–5)
    # MAC : pousse fort sur 3–5
    wmac_modes = np.array([2, 2, 3, 2, 2], dtype=float)

    # Fréquences : renforce sur les modes supérieurs (qui coincent)
    wf_modes = np.array([3.5, 3.5, 3.5, 3.5, 3.5], dtype=float)  # petit boost du poids freq du
mode 1

    # Residual, r, scaled
    r_s = (lamda_m - lamda_cur) / np.maximum(lamda_cur, 1e-2)



    # Keep for tracking
    lamda[itr, :] = lamda_cur
```

```
# types_full = [
#     # Z1 : Ec1_1, Ew1_1, Ew2_1, Es1_1
#     'E_c','E_w','E_w','E_s',
#     # Z2 : Ec1_2, Ec2_2, Ew1_2, Es1_2
#     'E_c','E_c','E_w','E_s',
#     # Z3 : Ec1_3, Ew1_3, Ew2_3, Es1_3
#     'E_c','E_w','E_w','E_s',
#     # Z4 : Ec1_4, Ec2_4, Ew1_4, Es1_4
#     'E_c','E_c','E_w','E_s',
#     # Z5 : Ec1_5, Ec2_5, Ec3_5, Ec4_5, Ew1_5, Ew2_5, Es1_5
#     'E_c','E_c','E_c','E_c','E_w','E_w','E_s'
# ]

types_full = [
    # Z1 : Param[0..5]
    'm11S', 'm22S', 'm12S', 'f11W', 'f22W', 'f12W',
    # Z2 : Param[6..11]
    'm11S', 'm22S', 'm12S', 'f11W', 'f22W', 'f12W',
    # Z3 : Param[12..17]
    'm11S', 'm22S', 'm12S', 'f11W', 'f22W', 'f12W',
    # Z4 : Param[18..23]
    'm11S', 'm22S', 'm12S', 'f11W', 'f22W', 'f12W',
    # Z5 : Param[24..29]
    'm11S', 'm22S', 'm12S', 'f11W', 'f22W', 'f12W',
]


# --- Pilotage adaptatif multi-bandes (modes 1–2 et 3–5) ---

# Erreurs fréquence (relatives) et MAC diag
```

```
err_f = np.abs(lamda_m - lamda_cur) / np.maximum(lamda_m, 1e-12)
mac   = MAC_diag_assigned.copy()

# Baisse relative de J (sur 2 itérations) pour détecter la stagnation
rel_drop = 0.0
if len(J_star_list) >= 3:
    J_now  = J_star_list[-1]
    J_prev = J_star_list[-3]
    if J_prev > 0:
        rel_drop = (J_prev - J_now) / J_prev

# Seuils (ajuste au besoin)
need_relax_35 = (mac[2:].mean() < 0.65) or (rel_drop < 0.15)
need_relax_12 = (err_f[0] > 0.06 or mac[0] < 0.995) or (err_f[1] > 0.06 or mac[1] < 0.995)



# 2) Bornes absolues pour E, m_ii et f_ii (en MPa), relatives à E0 # k00 # m00 (theta[0, :])
E_lo_factor = 0.70   # -30 %
E_hi_factor = 1.30   # +30 %
eps = 1e-12

# 3) Trust-region locale (autour de la valeur courante)
#    Tu peux resserrer à (0.90, 1.10) si le solveur oscille.
trust_E = (0.70, 1.60)

# 4) Cap d'incrément par itération (en espace réduit)
if itr == 0:
    per_cap_E = 1.0   # premier saut libre
elif itr < 5:
    per_cap_E = 0.65
```

```
    else:
        per_cap_E = 0.3


    # 5) (Optionnel) booster un peu la priorité fréquence des 2 premiers modes si besoin
    # wf_modes_dyn = wf_modes.copy()
    # if 'need_relax_12' in globals() and need_relax_12:
    #     wf_modes_dyn[0] *= 1.20
    #     wf_modes_dyn[1] *= 1.10

    wf_modes_dyn = wf_modes.copy()
    if itr < 3:
        wf_modes_dyn[0] *= 1.5
        if wf_modes_dyn.size > 1:
            wf_modes_dyn[1] *= 1.2

    # --- Caps physiques (MPa) ---
    # E_ABS_MIN = 1.75e4          # 17 500 MPa (filet bas)
    # E_CAP_MAX = 4.60e4          # 46 000 MPa (cap global dur)

    E_ABS_MIN = 0.1          # 0.1 (filet bas)
    E_CAP_MAX = 1.5          # 1.5 (cap global dur)

    # --- Cap haut par indice : ICI on force un cap GLOBAL à 46 000 pour tous
    theta0_ref = theta[0, :].astype(float)

    # Bornes "absolues" liées à E0 et cap global
    lo_abs_vec = np.maximum(0.70 * theta0_ref, E_ABS_MIN)
    hi_abs_vec = np.minimum(1.30 * theta0_ref, E_CAP_MAX)
```

```
# Valeur courante bornée dur (sécurité)
theta_cur = np.clip(theta[itr, :].astype(float), lo_abs_vec, hi_abs_vec)

if itr == 0:
    # scaling global 1ère passe
    scale = (lamda_m[0] / lamda_cur[0])**2
    lo0 = np.maximum(0.70 * theta0_ref, E_ABS_MIN)
    hi0 = np.minimum(1.30 * theta0_ref, E_CAP_MAX)      # << cap global ici
    theta[itr, :] = np.clip(theta[itr, :] * scale, lo0, hi0)
    theta_cur = theta[itr, :].astype(float)

lo_tr, hi_tr = (0.70, 1.60)

theta_min = np.empty_like(theta_cur)
theta_max = np.empty_like(theta_cur)

for i in range(theta_cur.size):
    lo_abs = max(0.70 * theta0_ref[i], E_ABS_MIN)
    hi_abs = min(1.30 * theta0_ref[i], E_CAP_MAX)   # ← combine les deux logiques

    lo_loc = lo_tr * theta_cur[i]
    hi_loc = hi_tr * theta_cur[i]

    theta_min[i] = max(lo_abs, lo_loc)
    theta_max[i] = min(hi_abs, hi_loc)

theta_max = np.maximum(theta_max, theta_min + 1e-12)

# Passage en espace réduit (rééchelonnage par E0)
dlo_s = (theta_min - theta_cur) / np.maximum(theta_cur, eps)
```

```
    dhi_s = (theta_max - theta_cur) / np.maximum(theta_cur, eps)

    # Poids fréquence dynamiques (cohérence A,b)
    W = np.diag(wf_modes_dyn.astype(float))

    A = -np.sqrt(W) @ G
    b = -np.sqrt(W) @ r_s
    lam = 1e-1
    A_ext = np.vstack([A, np.sqrt(lam)*np.eye(A.shape[1])])
    b_ext = np.concatenate([b, np.zeros(A.shape[1])])

    res = lsq_linear(A_ext, b_ext, bounds=(dlo_s, dhi_s), method='trf',
                     lsq_solver=None, lsmr_tol='auto', verbose=1)

    d_theta_s = res.x.copy()
    # for i, t in enumerate(types_full):
    #     cap = per_cap_mf if t == 'm_f' else per_cap_ff
    #     d_theta_s[i] = np.clip(d_theta_s[i], -cap, cap)

    # Cap uniforme (E uniquement)
    step_gain = 4   # essaie 1.5 à 2.0
    d_theta_s = step_gain * res.x

    # per_cap_E déjà défini plus haut (ex. 1.0 / 0.65 / 0.30)
    d_theta_s = np.clip(d_theta_s, -per_cap_E, per_cap_E)

    theta_prop = theta_cur + (d_theta_s * theta_cur)

    d_theta_upd = d_theta_s * theta_cur       # ΔE en MPa (pas relatif appliqué à la valeur
courante)
```

```
# Clamp bornes locales
theta_prop = np.minimum(np.maximum(theta_prop, theta_min), theta_max)

# Filets finaux 100 % durs (ne laissent RIEN dépasser 46 000)
theta_next = np.clip(theta_prop, E_ABS_MIN, E_CAP_MAX)  # << cap global final

# Écriture
theta[itr+1, :] = theta_next

# (Optionnel) sécurité numérique
assert np.all(theta[itr+1, :] <= E_CAP_MAX + 1e-9), "Cap 46 000 MPa violé"

# ------------------------- #
# UPDATE
# ------------------------- #



# ---------------------------------- #
# FEM ANALYSIS
# ---------------------------------- #

# ---------------- #
# EXPORT VARIABLES
# ---------------- #

Param = theta[itr+1]
# Ec1_1 = Param[0] #MPa,
# Ew1_1 = Param[1] #MPa,
# Ew2_1 = Param[2] #MPa,
# Es1_1 = Param[3] #MPa,
```

```
m11S_1 = Param[0]
m22S_1 = Param[1]
m12S_1 = Param[2]
f11W_1 = Param[3]
f22W_1 = Param[4]
f12W_1 = Param[5]

# Ec1_2 = Param[4] # MPa
# Ec2_2 = Param[5]
# Ew1_2 = Param[6]
# Es1_2 = Param[7]

m11S_2 = Param[6]
m22S_2 = Param[7]
m12S_2 = Param[8]
f11W_2 = Param[9]
f22W_2 = Param[10]
f12W_2 = Param[11]

# Ec1_3 = Param[8] #MPa,
# Ew1_3 = Param[9] #MPa,
# Ew2_3 = Param[10] #MPa,
# Es1_3 = Param[11] #MPa,

m11S_3 = Param[12]
m22S_3 = Param[13]
m12S_3 = Param[14]
f11W_3 = Param[15]
f22W_3 = Param[16]
```

```
f12W_3 = Param[17]

# Ec1_4 = Param[12] #MPa,
# Ec2_4 = Param[13] #MPa,
# Ew1_4 = Param[14] #MPa,
# Es1_4 = Param[15] #MPa,

m11S_4 = Param[18]
m22S_4 = Param[19]
m12S_4 = Param[20]
f11W_4 = Param[21]
f22W_4 = Param[22]
f12W_4 = Param[23]

# Ec1_5 = Param[16] #MPa,
# Ec2_5 = Param[17] #MPa,
# Ec3_5 = Param[18] #MPa,
# Ec4_5 = Param[19] #MPa,
# Ew1_5 = Param[20] #MPa,
# Ew2_5 = Param[21] #MPa,
# Es1_5 = Param[22] #MPa,

m11S_5 = Param[24]
m22S_5 = Param[25]
m12S_5 = Param[26]
f11W_5 = Param[27]
f22W_5 = Param[28]
f12W_5 = Param[29]
```

```
    # ret = SapModel.PropMaterial.SetMPIsotropic('80Conccol_5', Ec1_5*1000, 0.2, 0.0000099)
    # ret = SapModel.PropMaterial.SetMPIsotropic('CONC_5', Ec2_5*1000, 0.2, 0.0000099)
    # ret = SapModel.PropMaterial.SetMPIsotropic('45ConcCol_5', Ec3_5*1000, 0.2, 0.0000099)
    # ret = SapModel.PropMaterial.SetMPIsotropic('CONC30_5', Ec4_5*1000, 0.2, 0.0000099)

    # ret = SapModel.PropMaterial.SetMPIsotropic('FW_CONC60_5', Ew1_5*1000, 0.2, 0.0000099)
    # ret = SapModel.PropMaterial.SetMPIsotropic('FW_CONC35_5', Ew2_5*1000, 0.2, 0.0000099)

    # ret = SapModel.PropMaterial.SetMPIsotropic('CONC_S_5', Es1_5*1000, 0.2, 0.0000099)


    # PENSER à METTRE TOUS LES MODIFIERS DE LA ZONE ÉTUDIÉE AU DEBUT
    # + S'ASSURER QUE CEUX DES AUTRES ZONES ONT LES BONS
    Value = np.ones((10))

    Value[0] = 0.35
    Value[1] = 0.35
    Value[2] = 0.35
    Value[3] = m11S_5
    Value[4] = m22S_5
    Value[5] = m12S_5
    ret = SapModel.PropArea.SetModifiers("DALLE200_5", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DALLE225_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DALLE250_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DALLE300_5", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DALLE325_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DALLE350_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DALLE400_5", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DALLE450_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DROP250_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DROP350_5", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DROP450_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DROP475_5", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_5
Value[4] = m22S_5
Value[5] = m12S_5
ret = SapModel.PropArea.SetModifiers("DROP50_5", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("FW1", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("FW2", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("FW3", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW1", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW10", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW2", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW3", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW4", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW5", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW6", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW7", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW8", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW9", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("W1", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_5
Value[1] = f22W_5
Value[2] = f12W_5
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("W2", Value.tolist())
```

```
    # ret = SapModel.PropMaterial.SetMPIsotropic('60ConcCol_4', Ec1_4*1000, 0.2,
0.0000099)
    # ret = SapModel.PropMaterial.SetMPIsotropic('80Conccol_4', Ec2_4*1000, 0.2,
0.0000099)
    # ret = SapModel.PropMaterial.SetMPIsotropic('CONC60_W_4', Ew1_4*1000, 0.2,
0.0000099)
    # ret = SapModel.PropMaterial.SetMPIsotropic('CONC_S_4', Es1_4*1000, 0.2,
0.0000099)

    Value = np.ones((10))

    Value[0] = 0.35
    Value[1] = 0.35
    Value[2] = 0.35
    Value[3] = m11S_4
    Value[4] = m22S_4
    Value[5] = m12S_4
    ret = SapModel.PropArea.SetModifiers("DALLE250_4", Value.tolist())

    Value = np.ones((10))

    Value[0] = 0.35
    Value[1] = 0.35
    Value[2] = 0.35
    Value[3] = m11S_4
    Value[4] = m22S_4
    Value[5] = m12S_4
    ret = SapModel.PropArea.SetModifiers("DROP450_4", Value.tolist())
```

```
Value = np.ones((10))

Value[0] = f11W_4
Value[1] = f22W_4
Value[2] = f12W_4
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW19_4", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_4
Value[1] = f22W_4
Value[2] = f12W_4
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW18_4", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_4
Value[1] = f22W_4
Value[2] = f12W_4
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW17_4", Value.tolist())
```

```
    Value = np.ones((10))

    Value[0] = f11W_4
    Value[1] = f22W_4
    Value[2] = f12W_4
    Value[3] = 0.1
    Value[4] = 0.1
    Value[5] = 0.1
    ret = SapModel.PropArea.SetModifiers("SW20_4", Value.tolist())

    # ret = SapModel.PropMaterial.SetMPIsotropic('60ConcCol_3', Ec1_3*1000, 0.2,
0.0000099)
    # ret = SapModel.PropMaterial.SetMPIsotropic('CONC60_W_3', Ew1_3*1000, 0.2,
0.0000099)
    # ret = SapModel.PropMaterial.SetMPIsotropic('CONC45_W_3', Ew2_3*1000, 0.2,
0.0000099)
    # ret = SapModel.PropMaterial.SetMPIsotropic('CONC_S_3', Es1_3*1000, 0.2,
0.0000099)


    Value = np.ones((10))

    Value[0] = 0.35
    Value[1] = 0.35
    Value[2] = 0.35
    Value[3] = m11S_3
    Value[4] = m22S_3
    Value[5] = m12S_3
    ret = SapModel.PropArea.SetModifiers("DALLE250_3", Value.tolist())
```

```
    Value = np.ones((10))

    Value[0] = 0.35
    Value[1] = 0.35
    Value[2] = 0.35
    Value[3] = m11S_3
    Value[4] = m22S_3
    Value[5] = m12S_3
    ret = SapModel.PropArea.SetModifiers("DROP450_3", Value.tolist())

    Value = np.ones((10))

    Value[0] = f11W_3
    Value[1] = f22W_3
    Value[2] = f12W_3
    Value[3] = 0.1
    Value[4] = 0.1
    Value[5] = 0.1
    ret = SapModel.PropArea.SetModifiers("SW16_3", Value.tolist())

    # ret = SapModel.PropMaterial.SetMPIsotropic('CONC_2', Ec1_2*1000, 0.2, 0.0000099)
    # ret = SapModel.PropMaterial.SetMPIsotropic('CONC45_2', Ec2_2*1000, 0.2,
0.0000099)
    # ret = SapModel.PropMaterial.SetMPIsotropic('CONC45_W_2', Ew1_2*1000, 0.2,
0.0000099)
    # ret = SapModel.PropMaterial.SetMPIsotropic('CONC_S_2', Es1_2*1000, 0.2,
0.0000099)

    Value = np.ones((10))
```

```
Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_2
Value[4] = m22S_2
Value[5] = m12S_2
ret = SapModel.PropArea.SetModifiers("DALLE250_2", Value.tolist())

Value = np.ones((10))

Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_2
Value[4] = m22S_2
Value[5] = m12S_2
ret = SapModel.PropArea.SetModifiers("DROP450_2", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_2
Value[1] = f22W_2
Value[2] = f12W_2
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW15_2", Value.tolist())

Value = np.ones((10))
```

```
Value[0] = f11W_2
Value[1] = f22W_2
Value[2] = f12W_2
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW13_2", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_2
Value[1] = f22W_2
Value[2] = f12W_2
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW16_2", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_2
Value[1] = f22W_2
Value[2] = f12W_2
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW14_2", Value.tolist())
```

```
    # ret = SapModel.PropMaterial.SetMPIsotropic('CONC30_1', Ec1_1*1000, 0.2, 0.0000099)
    # ret = SapModel.PropMaterial.SetMPIsotropic('CONC_W', Ew1_1*1000, 0.2, 0.0000099)
    # ret = SapModel.PropMaterial.SetMPIsotropic('CONC45_W_1', Ew2_1*1000, 0.2, 0.0000099)
    # ret = SapModel.PropMaterial.SetMPIsotropic('CONC_S_1', Es1_1*1000, 0.2, 0.0000099)

    Value = np.ones((10))

    Value[0] = 0.35
    Value[1] = 0.35
    Value[2] = 0.35
    Value[3] = m11S_1
    Value[4] = m22S_1
    Value[5] = m12S_1
    ret = SapModel.PropArea.SetModifiers("DALLE250_1", Value.tolist())

    Value = np.ones((10))

    Value[0] = 0.35
    Value[1] = 0.35
    Value[2] = 0.35
    Value[3] = m11S_1
    Value[4] = m22S_1
    Value[5] = m12S_1
    ret = SapModel.PropArea.SetModifiers("DALLE225_1", Value.tolist())

    Value = np.ones((10))
```

```
Value[0] = 0.35
Value[1] = 0.35
Value[2] = 0.35
Value[3] = m11S_1
Value[4] = m22S_1
Value[5] = m12S_1
ret = SapModel.PropArea.SetModifiers("DROP450_1", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_1
Value[1] = f22W_1
Value[2] = f12W_1
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW22_1", Value.tolist())

Value = np.ones((10))

Value[0] = f11W_1
Value[1] = f22W_1
Value[2] = f12W_1
Value[3] = 0.1
Value[4] = 0.1
Value[5] = 0.1
ret = SapModel.PropArea.SetModifiers("SW21_1", Value.tolist())

Value = np.ones((10))
```

```
    Value[0] = f11W_1
    Value[1] = f22W_1
    Value[2] = f12W_1
    Value[3] = 0.1
    Value[4] = 0.1
    Value[5] = 0.1
    ret = SapModel.PropArea.SetModifiers("SW15_1", Value.tolist())

    Value = np.ones((10))

    Value[0] = f11W_1
    Value[1] = f22W_1
    Value[2] = f12W_1
    Value[3] = 0.1
    Value[4] = 0.1
    Value[5] = 0.1
    ret = SapModel.PropArea.SetModifiers("SW13_1", Value.tolist())

    ret = SapModel.File.Save('Z:\\Cyrielle\\4. Mémoire\\FINAL FINAL MODEL USED FOR
FINAL UPDATING\\CALL\\700SJ ELS stage construction final CALIBRATION1.EDB')

    # Parameter list
    np.save('parameter_list_upd' + '.npy', theta[itr+1, :], allow_pickle=True,
            fix_imports=True)

    # ---------------- #
    # RUN ANALYSIS
    # ---------------- #

    ret = SapModel.Analyze.RunAnalysis()
```

```
ModalTable = []
ModeShape = []
ModalTableFINAL = []
ModeShapeFINAL = []

#Exécuter l'analyse modale
SapModel.Analyze.RunAnalysis()

ret = SapModel.Results.Setup.SetCaseSelectedForOutput("MODAL")
ret = SapModel.Results.Setup.SetOptionModeShape(1, 15, True)

NumberResults = 0
Obj = ""
Elm = ""
LoadCase = "MODAL"
StepType = ""
StepNum = []
U1 = []
U2 = []
U3 = []
R1 = []
R2 = []
R3 = []

TABLE = SapModel.Results.JointDispl("ALL", 2, NumberResults, Obj, Elm, LoadCase, StepType, StepNum, U1, U2, U3, R1, R2, R3)
TABLE = TABLE[5:8]
x = np.array(TABLE[0])
y1 = np.array(TABLE[1])
```

```
    y2 = np.array(TABLE[2])

    valeurs_uniques = np.unique(x)

    # Résultat : liste de tuples (valeur x, max abs y1, max abs y2)
    resultats = []

    for val in valeurs_uniques:
        indices = np.where(x == val)[0]
        max_y1 = np.max(np.abs(y1[indices]))
        max_y2 = np.max(np.abs(y2[indices]))
        resultats.append((val, max_y1, max_y2))

    ret = SapModel.Results.JointDispl("7442", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7444", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7445", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7446", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)
```

```
    ret = SapModel.Results.JointDispl("7447", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7448", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7451", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7449", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7450", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7454", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7455", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7458", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
```

```
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7459", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7461", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7462", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7463", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7464", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7465", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7466", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)
```

```
    ret = SapModel.Results.JointDispl("7467", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7471", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7472", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7473", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7474", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7475", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7476", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7477", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
```

```
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7478", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7479", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7480", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7481", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7482", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7483", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7484", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)
```

```
    ret = SapModel.Results.JointDispl("7485", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7486", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7487", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7488", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7489", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7490", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7491", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7492", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
```

```
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7493", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7494", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7495", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7496", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7497", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7531", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7499", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)
```

```
    ret = SapModel.Results.JointDispl("7500", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7501", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7502", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7503", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7504", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7505", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7506", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7507", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
```

```
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7508", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7509", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7510", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7511", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)

    ret = SapModel.Results.JointDispl("7513", 1, NumberResults, Obj, Elm, LoadCase,
StepType, StepNum, U1, U2, U3, R1, R2, R3)
    ModeShape.append(ret)



    ModeShapeFINAL.append(ModeShape.copy())
    ModeShapeITE = ModeShapeFINAL[0]

    Ux = []
    for i in range(len(ModeShapeITE)):
        Ux.append(ModeShapeITE[i][6])
```

```
Ux = np.array(Ux)

for i in range(20):
    Ux[:,i] = Ux[:,i]/resultats[i][1]

Uy = []
for i in range(len(ModeShapeITE)):
    Uy.append(ModeShapeITE[i][7])

Uy = np.array(Uy)

for i in range(20):
    Uy[:,i] = Uy[:,i]/resultats[i][2]

modes_num_upd = np.vstack((Ux, Uy))
modes_num_upd = modes_num_pt[:, :5]
modes_num_upd = mass_normalize(modes_num_pt,Mass_vec)

ret = SapModel.Results.Setup.SetCaseSelectedForOutput("MODAL")

# Initialiser les variables
NumberResults = 0  # Cette variable sera remplie par la méthode ModalPeriod()

# Les arrays doivent être de type spécifique. Ici, nous utilisons array de type 'd' pour les doubles et 'u' pour les strings
LoadCase = ""
StepType = ""
StepNum = []
Period = []
Frequency = []
```

```
    CircFreq = []
    EigenValue = []

    ret = SapModel.Results.ModalPeriod(NumberResults, LoadCase, StepType, StepNum,
Period, Frequency, CircFreq, EigenValue)
    ModalTable.append(ret)
    ModalTableFINAL.append(ModalTable.copy())

    for i in range(5):
        Frequency.append(ModalTableFINAL[0][0][5][i])

    # Convertir en array colonne
    frequency_array = np.array(Frequency).reshape(-1, 1)
    mode_ids = np.arange(1, len(Frequency) + 1).reshape(-1, 1)
    freqs_num_upd = np.hstack((mode_ids, frequency_array))

    ret = SapModel.SetModelIsLocked(False)

    # ---------------------------------- #
    # POST-PROCESS
    # ---------------------------------- #

    # Sorted MAC and MMI
    (MAC_updated, _, results_MAC_upd, _) = get_MAC_MMI(freqs_est,
                                        modes_est,
                                        freqs_num_upd,
                                        modes_num_upd,
                                        filtering=False)

    # Updated frequencies
```

```
lamda_upd = np.zeros(q, dtype=float)
lamda_upd_all = np.zeros(cols_est, dtype=float)

for idx, mode in enumerate(mode_list):

    # Establish frequencies based on the mode_list string
    string_idx = mode[-2:]

    # Correct for 0 in the string
    if string_idx[0] == 0:

        f_idx = int(mode[-1]) - 1

    else:
        f_idx = int(string_idx) - 1

    # Updated numerical frequencies
    # Top match based on MAC
    lamda_upd[idx] = results_MAC_upd[mode][0, 2]

# All updated numerical frequencies
for idx, mode in enumerate(results_MAC_upd.keys()):

    lamda_upd_all[idx] = results_MAC_upd[mode][0, 2]

# Update lamda
lamda[itr+1, :] = np.copy(lamda_upd)
lamda_all[itr+1, :] = np.copy(lamda_upd_all)

# Append updated MAC results
```

```
MAC_updated_model.update({'MAC_updated' + str(itr+1): MAC_updated})
results_MAC_updated_model.update({'results_MAC_upd' + str(itr+1):
                                  results_MAC_upd})

# ------------------------- #
# OBJECTIVE FUNCTION
# ------------------------- #

# Sorted MAC/MMI sur les DONNÉES UPD
(MAC_updated, MMI_upd, results_MAC_upd, _) = get_MAC_MMI(
    freqs_est, modes_est,
    freqs_num_upd, modes_num_upd,
    filtering=False
)

# Hungarian pour réordonner le MAC (upd)
ri_upd, ci_upd = linear_sum_assignment(-MMI_upd)
perm_upd = np.argsort(ci_upd)
rows_upd = ri_upd[perm_upd]
MAC_assigned_upd = MAC_updated[rows_upd, :]
MAC_diag_upd = np.diag(MAC_assigned_upd)

# Fréquences réordonnées (upd)
freqs_num_assigned_upd = freqs_num_upd[rows_upd, :]
lamda_upd = freqs_num_assigned_upd[:, 1]

# Objective J* (upd)
wf   = 1.0
wmac = 0.5   # voire 0.5 au début
beta = 0.0   # commence SANS pénalité off-diag, on la remettra plus tard (ex. 0.02)
```

```
    off_upd = MAC_assigned_upd.copy()
    np.fill_diagonal(off_upd, 0.0)

    J_f   = np.sum(wf_modes * ((lamda_m - lamda_upd) / np.maximum(lamda_m, 1e-12))**2)
    J_mac = np.sum(wmac_modes * (1.0 - MAC_diag_upd)**2) + beta*np.sum(off_upd**2)
    J_star = wf*J_f + wmac*J_mac
    J_star_list.append(J_star)   # <- une seule fois
    print(f"Iter {itr}: J* = {J_star:.4f}, J_f = {J_f:.4f}, J_mac = {J_mac:.4f}")



# plt.figure(figsize=(8, 5))
# plt.plot(iterations, J_star_list, marker='o', linestyle='-', linewidth=2)
# plt.title("Évolution de la fonction objectif $J^*$", fontsize=14)
# plt.xlabel("Itération", fontsize=12)
# plt.ylabel("$J^*$", fontsize=12)
# plt.grid(True)
# plt.tight_layout()
# plt.show()

# ------------------------------------------------------------------------ #
# MOVE FILES
# ------------------------------------------------------------------------ #



# ------------------------------------------------------------------------ #
# EXPORT
# ------------------------------------------------------------------------ #
```

```
# Export relevant variables to postprocessing
info = {'freqs_est': freqs_est,
        'modes_est': modes_est,
        'freqs_num': freqs_num_upd,
        'modes_num': modes_num_upd,
        'theta': theta,
        'MAC_initial': MAC_initial_model,
        'MAC_updated': MAC_updated_model,
        'lamda_m': lamda_m,
        'lamda': lamda,
        'lamda_all': lamda_all,
        'G_matrices': G_matrices,
        'W': W,
        'J_star_list': J_star_list}

with open("info.txt", "w") as file:
    for key, value in info.items():
        file.write(f"{key} = {value}\n")



# ------------------------------------------------------------------------ #
# END OF ANALYSIS
# ------------------------------------------------------------------------ #
```

## LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES


Bonne, Alexis. 2018. «Caractérisation de l'effet des composants non structuraux sur les propriétés dynamiques des bâtiments.» *Mémoire de maitrise* . Montréal: École de Technologie Supérieure.

Boscato, Giosuè, Salvatore Russo, Rosario Ceravolo, et Luca Zanotti Fragonara. 2015. *Global Sensitivity-Based Model Updating.* 20 Janvier.

Brincker, Rune, et Carlos E Ventura. 2015. *Introduction to Operational Modal Analysis.* United Kingdom: John Wiley & Sons, Ltd.

Broccolini. 2024. «Victoria sur Le Parc.»

Brownjohn, J.M.W. 2003. «Ambient vibration studies for system identification of tall buildings.» Janvier.

CanadaMaps360. 2016. «Canada cities map.»

Chopra, Anil K. 2014. *Dynamics of structure 4th edition.* Pearson.

CNBC - Division 4. 2020.

Computers & Structures America, Inc. 2014. Accessed April 4, 2014.

Computers & Structures, Inc. 2025. *ETABS Building Analysis and design.*

Council on Tall Buildings and Urban Habitat. 2024. «Republic Plaza.»

CSA Group - A23.3-14. 2014. "Design of concrete structures."

CTBUH. 2023. *Critères des bâtiments de grande hauteur.*

Dynamic Design Solutions. 2025. «FEMtools.»

Edmund Tie. 2024. «The JTC Summit.» *Edmind Tie.*

Gouvernement du Canada. 2008. *Publications du Gouvernement du Canada.*

Guide de l'utilisateur - CNB 2015 : partie 4 de la division B. 2015.

Hu, Jun, Heung-Fai Lam, et Jia-Hua Yang. 2018. *Operational modal identification and finite element model updating of a coupled building following Bayesian approach.* Février.

Hurtado, Oscar D, Albert R Ortiz, Daniel Gomez, and Rodrigo Astroza. 2023. "Bayesian Model-Updating Implementation in a Five-Story Building." Juin 20.

Li, Bing, Graham L. Hutchinson, and Colin F. Duffield. 2011. *The influence of non-structural components on tall building stiffness.* Novembre 03.

Li, S.Y, M. Liu, H.X. Li, Y. Hui, et Z.Q Chen. 2017. «Effects of structural damping on wind-induced responses of a 243-meter-high solar tower based on a novel elastic test model.» 15 Novembre.

Li, Xiang, Carlos E. Ventura, Yu Feng, Yuxin Pan, Yavuz Kaya, Haibei Xiong, Fengliang Zhang, Jixing Cao, et Minghui Zhou. 2016. «Ambient Vibration Testing of Two Highly Irregular Tall Buildings in Shanghai.» May.

Magalhães, Filipe, Alvaro Cunha, et Elsa De Sa Caetano. 2012. *Vibration based structural health monitoring of an arch bridge: From automated OMA to damage detection.* Avril.

Moayedi, Ferya, Salman Soleimani-Dashtaki, et Carlos E. Ventura. 2015. «Determination of Modal Properties of an Irregular 20-Story Concrete Shear Wall Building.»

MoHo. 2025.

Moho Science and Technology. 2025.

Morri. 2024. *Shanghai World Financial Center.*

Mottershead, John E., Michael Link, et Michael I. Friswell. 2011. *The sensitivity method in finite element model updating: A tutorial.* Octobre.

Pan, Yuxin, Carlos Ventura, Haibei Xiong, et Feng-Liang Zhang. 2020. «Model updating and seismic response of a super tall building in Shanghai.» 18 Juin.

Peeters, Bart, et Guido De Roeck. 2001. *Stochastic System Identification for Operational Modal Analysis: A Review.* Décembre.

Peng, Liu, Cheng Yu, et Zhu Yan-Song. 2016. *The Structural Design of China Zun Tower, Beijing.* September.

Razavi, Saman, et Hoshin V Gupta. 2015. «AGU Publications.» *What do we mean by sensitivity analysis? The need for comprehensive characterization of "global" sensitivity in Earth and Environmental systems models.* 23 Mai.

REW. 2024. «The Melville.»

Reynders, Edwin. 2012. *System Identification Methods for (Operational) Modal Analysis: Review and Comparison.* 9 Février.

Shi, Weixing, Jiazeng Shan, et Xilin Lu. 2011. «Modal identification of Shanghai World Financial Center both from free and ambient vibration response.» *Engineering Structures.*

Singh, Ashish, et Sasankasekhar Mandal. 2023. «Effect of wind and structural parameters on across wind load of super high-rise buildings.» *Research on Engineering Structures & Materials.*

Siu-Kui, Au. 2011. «Fast Bayesian ambient modal identification in the frequency domain, Part I: Posterior most probable value.» *Mechanical Systems and Signal Processing.*

Smith, Rob, et Michael Willford. 2008. «Damping in tall buildings – uncertainties and solutions.» *IABSE Chicago 2008 conference.*

Svendsen, Bjørn T, Øyvind W Petersen, Gunnstein T Frøseth, et Anders Rønnquist. 2021. *Improved finite element model updating of a fullscale steel bridge using sensitivity analysis.* 15 Juillet.

SVIBS. 2025. *Manual Harmonic Indicators.*

—. 2025. *Enhanced Frequency Domain Decomposition.*

—. 2025. *Modal Assurance Criterion.*

—. 2025. *Modal Estimation window - Stochastic Subspace Identification.*

—. 2025. *Signal Processing Window.*

Teughels, Anne, J. Maeck, et Guido De Roeck. 2002. *Damage assessment by FE model updating using damage functions.* Septembre.

Tian, Wei. 2018. «A review of sensitivity analysis methods in building energy analysis.» 22 Juin.

TROMINO. 2025.

Turek, Martin, Carlos E. Ventura, and Sebastián Guerrero. 2007. "Ambient Vibration Testing and Model Updating of a 44-Storey." Janvier.

Turek, Martin, Carlos E. Ventura, et Sebastián Guerrero. 2007. «Ambient Vibration Testing and Model Updating of a 44-Storey.» Janvier.

Wang, Meng, Satish Nagarajaiah, et Fei-Fei Sun. 2022. «A novel crosswind mitigation strategy for tall buildingsusing negative stiffness damped outrigger systems.» 19 April.

Wu, J.R, et Q.S Li. 2004. *Finite element model updating for a high-rise structure based on ambient vibration measurements.* 2 Mars.

Yuan, Zhaoxu, Peng Liang, Tiago Silva, Kaiping Yu, et John E Mottershead. 2018. «Parameter selection for model updating with global sensitivity analysis.» *Mechanical Systems and Signal Processing.* 15 Juin.

Yun, Da Yo, Doyoung Kim, Minsun Kim, Sang Geun Bae, Jae Woo Choi, Hak Bo Shim, Taehoon Hong, Dong-Eun Lee, et Hyo Seon Park. 2020. «Field measurements for identification of modal parameters for high-rise.» 28 Octobre.

Zhang, Feng-Liang, Carlos E. Ventura, Hai-Bei Xiong, Wen-Sheng Lu, Yu-Xin Pan, et Ji-Xing Cao. 2017. «Evaluation of the dynamic characteristics of a super tallbuilding using data from ambient vibration and shake tabletests by a Bayesian approach.» 11 Octobre.