Analyse modale opérationnelle appliquée à la caractérisation dynamique in situ d'une dalle de ventilation

Steve Carrier

UNIVERSITÉ DE LYON

ENTPE

L'école de l'aménagement durable des territoires

Travail de fin d'etudes

pour le diplôme d'ingenieur de l'Ecole Nationale des Travaux Publics de l'Etat

---

Année : 2018-2019
Soutenu le 26 juin 2019

Voie d'approfondissement : Génie civil

Devant le jury composé de :

- Président du jury : Claude BOUTIN
- Maître de TFE : Antoine RALLU
- Expert : Alireza TURE SAVADKOOHI
- Invité : Stéphane HANS

Par

Steve CARRIER

Analyse modale opérationnelle appliquée à la caractérisation dynamique in situ d'une dalle de ventilation

Centre d'études des tunnels (CETU)

CETU

Notice analytique

AUTEUR	
Nom	CARRIER
Prénom	Steve
ORGANISME D'ACCUEIL	
Nom de l'organisme et localité	Centre d'études des tunnels (CETU à Bron)
Nom du tuteur	Antoine RALLU
ANALYSE DU TFE	
Titre	Analyse modale opérationnelle appliquée à la caractérisation dynamique in situ d'une dalle de ventilation
Title	Operational modal analysis applied to the dynamic in situ characterization of a ventilation slab
Résumé	Le CETU souhaite développer ses compétences en auscultation dynamique d'ouvrages. Pour cela, nous mettons en pratique différentes techniques de traitements de signaux afin de détecter l'endommagement ou non d'une dalle de ventilation de tunnel. C'est pourquoi, nous développons différentes techniques d'analyses modales opérationnelles. Deux techniques de traitement du signal sont particulièrement développées, la 'Frequency Domain Decomposition' et les modèles de Prony. Une analyse critique des résultats obtenus et des différentes méthodes utilisées est faite.
Abstract	CETU wishes to develop its skills in dynamic auscultation of structures. For this, we put into practice different signal processing techniques in order to detect the damage or not of a tunnel ventilation slab. This is why we are developing different operational modal analysis techniques. Two signal processing techniques are particularly developed, the Frequency Domain Decomposition and the Prony models. A critical analysis of the results obtained and the different methods used is made.
Mots clés	Analyse modale opérationnelle, Domaine de décomposition fréquentiels, Modèles de Prony, procédure auto-régressives, poutre de flexion
Keywords	Operationnal Modal Analysis, Frequency Domain Decomposition, Prony's Model, Auto-regressive model, bending beam
Termes géographiques	Tunnel du Siaix en Savoie
COLLATION	
Nombre de pages	
Nombre d'annexes	
Nombre de références bibliographiques	19

Remerciements

Je tiens tout d'abord à remercier Antoine RALLU, mon maître de stage. Tout d'abord, pour m'avoir accueilli en stage malgré la durée limité de celui-ci. Je le remercie ensuite pour le soutien qu'il m'a apporté tout au long de mon stage et de l'exigence qu'il a eu envers moi. Je remercie ensuite le CETU pour m'avoir accueilli en stage et m'avoir fourni le matériel nécessaire à la réalisation de mon TFE.

Je remercie ensuite le laboratoire du génie civil et du bâtiment de nous avoir prêtés les différents capteurs. Ceux-ci nous ont permis de recueillir un maximum d'informations. Ces informations furent la base de mon travail durant mon TFE.

Je remercie également monsieur Ture Savadkoohi d'avoir accepté d'être l'expert lors de ma soutenance de TFE et ainsi de consacrer de son temps pour évaluer mon travail. De plus je remercie monsieur Boutin d'être le président de mon jury et ainsi d'évaluer mon travail.

Table des matières

Table des figures

Introduction

La pérennité des structures de génie civil est un sujet d'intérêt public. Notamment les infrastructures tels que les ponts et tunnels qui contribuent à l'économie de la société. Ces structures sont faites pour durer, perdurer à travers le temps. Cependant, un ouvrage est sans cesse exposé à des menaces, des agressions de l'environnement extérieur. Par exemple lors de travaux à proximité d'ouvrages existants il y a toujours un risque de détérioration de l'état de ces structures avoisinantes. Ces agressions peuvent avoir des effets sur la vie de l'ouvrage et à fortiori sur son exploitation (un pont qui serait en trop mauvais état devra être fermé pour éviter une catastrophe). Ainsi il est nécessaire de s'assurer de bon état d'un ouvrage.

Plusieurs techniques existent pour contrôler l'état d'un ouvrage. Le contrôle peut être effectué de manière visuel dans un premier temps. Par exemple une armature qui deviendrait visible (à cause d'une détérioration du béton l'enrobant) dans une structure en béton armé serait un indicateur sur l'état de l'ouvrage. Ensuite des techniques d'auscultation sont mises en place pour vérifier l'état des structures. Les techniques d'auscultation destructrices en sont première catégorie. Ces techniques consistent à prélever des échantillons sur place puis à les soumettre à divers tests mécaniques en laboratoire. Ces tests sont destructifs, c'est à dire que l'on va démolir l'échantillon. Ainsi ces tests suppriment des parties de l'ouvrage. De plus ces tests vont permettre de caractériser l'état de la structure en un point localisé (à l'endroit où l'échantillon fut prélevé).

Il existe cependant d'autres techniques d'auscultations des ouvrages qui ne sont pas destructives, appelées passives. En particulier, l'analyse dynamique des structures (mesures des vibrations) permet de caractériser l'état global d'une structure sans être destructif. C'est pourquoi ces techniques d'analyses se sont répandues à partir des années 1990 et se sont largement développées dans le génie civil. Elle s'y applique dans beaucoup de secteurs comme les plates-formes offshores, les bâtiments, en passant par les ponts et les gratte-ciels.

Cependant il existe plusieurs façon d'analyser une structure dynamiquement. L'analyse dynamique consiste à caractériser dynamiquement (in situ) une structure par ses modes de vibrations. On définit un mode comme étant un ensemble de trois éléments une fréquence propre, un amortissement et une déformée (triplet $[f, \xi$,déformée]). En effet chaque structure 'vibre' à certaines fréquences qui lui sont propres. Les fréquences propres d'une structure dépendent de ses caractéristiques géométriques intrinsèques(longueur, largeur, inertie...), ses propriétés mécaniques (module d'élasticité, masse volumique...) et des conditions aux limites de celle-ci.

Néanmoins pour qu'une structure vibre il est nécessaire de la soumettre à une excitation. L'excitation peut être un choc, une oscillation mécanique forcée ou une excitation quelconque. En réalité une structure peut être vue comme un filtre linéaire qui ne va laisser passer que certaines fréquences. Si l'on connaît le signal d'entrée (c'est le cas pour des chocs et des oscillations mécaniques forcées) et le signal de sortie il est possible de déterminer les modes de la structure. On parle alors d'analyse modale expérimentale (experimental modal analysis : EMA). Ce type d'analyse est très répandue en industrie pour des petits éléments facilement excitables. Or dans le génie civil l'excitation 'artificielle' d'une structure est très coûteuse car elle nécessite des systèmes assez puissants pour exciter la structure. C'est pourquoi l'EMA n'est pas très répandue en génie civil. En revanche une autre technique d'analyse modale est elle très répandue dans ce secteur, il s'agit de l'analyse modale opérationnelle (Operational modal analysis : OMA). Elle consiste à mesurer les vibrations sous une excitation ambiante de la structure (bruit des voitures, excitation

du vent...). En OMA on mesure seulement la sortie du filtre (le filtre étant la structure), ainsi on fait une hypothèse sur le signal d'entrée du filtre. On suppose que le signal d'entrée est un bruit blanc. Un bruit blanc est un signal pour lequel toutes les fréquences ont la même puissance.

Après avoir mesuré le signal de sortie, il faut le traiter pour en dégager des informations. Les informations que l'on cherche à récupérer sont celles propres aux modes. C'est à dire que par rapport au signal de sortie nous cherchons à récupérer les modes de la structure. Il existe plusieurs méthodes de traitement du signal. Chaque méthode à ses intérêts et ses inconvénients. Deux catégories de traitement du signal sont particulièrement répandues, le traitement fréquentiel du signal et le traitement du signal dans le domaine temporel. Dans la première catégorie, les données du signal sont directement traitées dans le domaine des fréquences par l'utilisation de la transformée de Fourier. La seconde méthode consiste à traiter le signal qui se situe dans le domaine temporel dans un premier temps puis de le transférer dans le domaine fréquentiel pour déterminer les modes du système.

L'OMA permet d'avoir des informations sur la structure, qu'il faut interpréter pour la caractériser. Mais comment caractériser l'état de la structure par rapport à ces informations ? L'idée est de comparer les informations recueillies à un instant t aux informations recueillies à un instant postérieur à celui-ci. Cependant à partir des ces comparaisons, il est possible de déterminer avec un certain intervalle de confiance si la structure est endommagée ou non.

Le rôle du Centre d'études des Tunnels est de "doter le ministère d'une compétence dans l'ensemble des techniques et méthodes relatives à l'exploitation et la sécurité des tunnels". Ainsi un des rôles de ce service est de développer des techniques d'inspection des ouvrages dont il a la charge. Dans le cadre de mon travail de fin d'études (TFE), nous nous intéressons à la dalle de ventilation du tunnel du Siaix en Savoie. Le tunnel en question a connu une série de travaux entre 2017 et 2019. Ils ont été réalisés suite à l'évolution des normes de sécurité, conséquence de l'incendie du Mont Blanc(1989). En effet, suite à cet accident tous les tunnels d'une longueur supérieure à cinq cents mètres doivent avoir une galerie de sécurité. Ces travaux consistaient à creuser des rameaux conduisant à la galerie de sécurité à différents points métriques, une inquiétude fut formulée quant à l'endommagement possible de la dalle. Dans les tunnels le passage d'ondes mécaniques endommage rarement la voûte mais il est courant que les autres éléments du tunnel le soit. L'objectif de mon TFE est d'utiliser ce cas pour mettre en application différentes méthodes liées à l'analyse modale opérationnelles afin de dégager les intérêts et limites de cette méthode pour qu'à terme son utilisation soit simplifiée.

Ce travail de fin d'études s'articule donc en trois parties. La première partie consiste à modéliser la structure de manière à déterminer les modes de celle-ci analytiquement. L'étude analytique portera sur une structure constitué de trois poutres de type Euler-Bernoulli (poutre de flexion) et sera comparée aux résultats d'un modèle numérique. La seconde pose les bases des différentes techniques de traitement du signal en analyse modale opérationnelle puis détaille l'application de deux techniques par rapport aux données prélevées sur la structure d'étude. Les deux techniques employées sont la 'Frequency Domain Decomposition' (FDD) et deux procédures paramétriques (procédures Auto-Régressive : AR ; procédure Auto-Régressive à moyenne mobile : ARMA). La FDD se base sur une technique fréquentielle de traitement du signal tandis que les procédures AR et ARMA cherchent à caractériser le signal temporel à partir d'un certains nombre de paramètres. Il existe différentes méthodes d'estimations de ces paramètres qui feront l'objet de discussion. Enfin la dernière partie traite des différentes méthodes de comparaisons qui vont être mises en place pour caractériser l'endommagement ou non de la structure. La première technique consiste à observer les changements des fréquences estimées à la partie précédente, d'une série de données à une autre. La seconde se base sur les méthodes de l'indicateur d'endommagement. Un indicateur est un vecteur de données déterminées. En l'occurrence, un des indicateurs sera les paramètres de la procédure AR à un instant t à comparer par rapport aux autres paramètres AR estimées. Pour enfin conclure sur les résultats (et limite de ces résultats) obtenus par ces différentes méthodes.

Première partie

Modélisation dynamique de la structure

Explications sur le choix du modèle

Le schéma de la dalle de ventilation du tunnel du Siaix est disponible en figure 1. Sur celui-ci on observe que la dalle est composé de trois poutres, deux horizontales (les poutres en béton) et une verticale la suspente métallique. La suspente métallique est modélisée par une poutre rectangulaire constituée d'un matériau homogène équivalent. Les trois poutres furent reliées par un coffrage en béton. Ainsi il fut décidé de modéliser ce coffrage par un encastrement. De plus la suspente est fixée en clé de voûte. Ce mécanisme empêche la rotation et le déplacement de la suspente il est donc raisonnable de modéliser cette fixation par un encastrement.

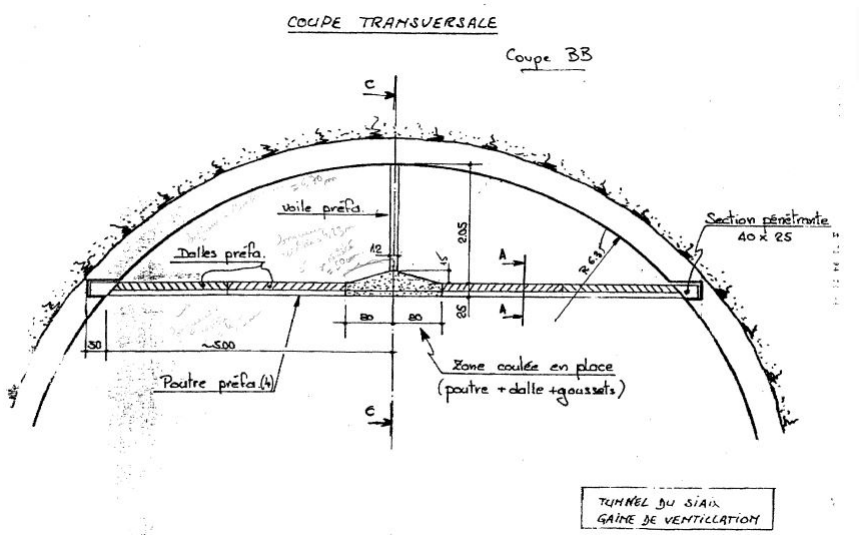

FIGURE 1 – Schéma de la dalle de ventilation réelle

Une discussion peut être menée sur le type de condition aux limites pour les poutres bétons. En effet on remarque qu'elles sont logées à l'intérieur d'une cavité qui leur est dédiée. A première vue il semblerait que ces poutres soient seulement posées dans cette cavité (c'est-à-dire qu'elle ne soient pas en contact horizontalement avec la voûte). Ainsi cette liaison entre la voûte (qui est considérée fixe) et les poutres bétons peut être modélisée par deux appuis simples.

Néanmoins il est possible qu'après avoir posé la dalle de ventilation des phénomènes de convergences (resserrement du tunnel) aient eu lieu. En effet, il s'agit d'un phénomène différé, dont les manifestations peuvent persister plusieurs mois après l'excavation. De plus le fait qu'il y ait eu de nouveaux travaux à proximité du tunnel pourrait induire ce genre de phénomène. Ainsi si la voûte se resserre il se peut qu'elle rentre en contact horizontalement avec la poutre. Si tel est le cas, cela signifierait que la poutre serait bloqué verticalement et horizontalement. Elle serait donc bloquée en déplacements verticaux, déplacements horizontaux et en rotation. Ainsi cette liaison ne pourrait plus être modélisée par deux appuis simples mais par deux encastrement.

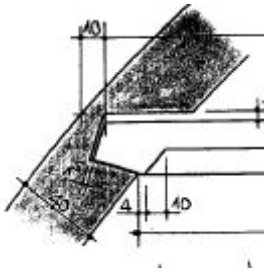

FIGURE 2 – Liaison poutre de la poutre béton avec la voûte

L'étude de la modélisation d'un encastrement de la suspente et des deux appuis simples des

poutres en béton fait l'objet de la section 2.1. L'étude du second modèle fait l'objet de la section 2.2.

Chapitre 1

Modèle dynamique de poutre de flexion

1.1 Hypothèses

1.1.1 Hypothèses liées au modèle

- On modélise la structure comme étant un ensemble de trois poutres (2 poutres de bétons et 1 poutre mixte acier-béton) dont la connexion est supposé encastrée. Les conditions aux limites du système seront dans un premier temps un encastrement de la poutre mixte (suspente) et deux appuis simples pour les poutres bétons. Dans un second temps on considérera trois encastrements (dûs à des phénomènes de convergence). Les explications de chacune de ces modélisations seront détaillées par la suite. Le schéma du premier modèle est présenté en figure 1.1.

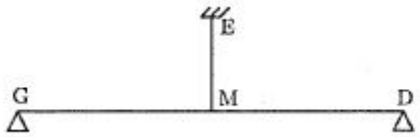

FIGURE 1.1 – Schéma premier modèle (le second modèle remplace les deux appuis simples par des encastrements)

- Lors de vibrations sous bruit ambiant **les déplacements sont de l'ordre du micromètre**. Ainsi, l'hypothèse des petites déformations dans le cadre de cette dynamique est plausible ;
- Puisque l'on considère des petites déformations, il est acceptable de considérer que **chaque élément a un comportement élastique linéaire**. On suppose également l'isotropie ;
- On considère **un modèle de poutre élancée**. Une poutre est dite élancée lorsque le diamètre équivalent est au moins 20 fois plus faible que la longueur de la poutre. Or le diamètre équivalent des poutres bétons est de 0,63 mètre pour une longueur de 5 mètres. Ainsi pour les éléments de poutres bétons considérées les longueurs sont biens supérieures aux diamètres équivalents. Un modèle de poutre élancée est donc acceptable (on vérifie également que les largeurs et hauteurs des poutres sont très faibles devant leurs longueurs) ;
- **On fait également l'hypothèse de Bernoulli** (qui est acceptable dans le cas de poutres élancées, en petites déformations). On suppose donc que lorsque des déformations se produisent, les sections droites restent droites par rapport à la courbe moyenne de la poutre.

Cette hypothèse implique que la rupture sera due à l'extension des fibres sur l'extérieur de la partie en flexion. Le cisaillement est donc négligé. Dans un modèle de Timoshenko le cisaillement serait pris en compte. ;

- L'étude est supposée être faite en **régime harmonique à une fréquence fixée**. Ainsi toutes les variables (déplacements, contraintes, forces...) peuvent être exprimées en tant que produit d'une fonction de l'espace et d'une fonction dépendante du temps.
- Les longueurs d'ondes sont supposées être très grandes devant l'épaisseur des poutres. C'est à dire que les longueurs d'ondes seront d'au moins 2,5 mètres (10 fois l'épaisseur des poutres bétons).

1.1.2 Caractéristiques géométriques et mécaniques des poutres

Poutres bétons

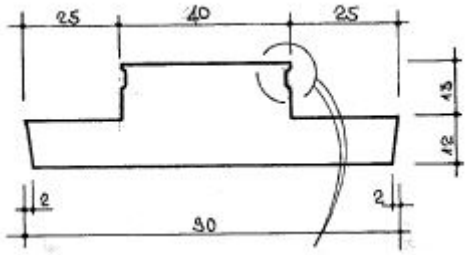

FIGURE 1.2 – coupe beton

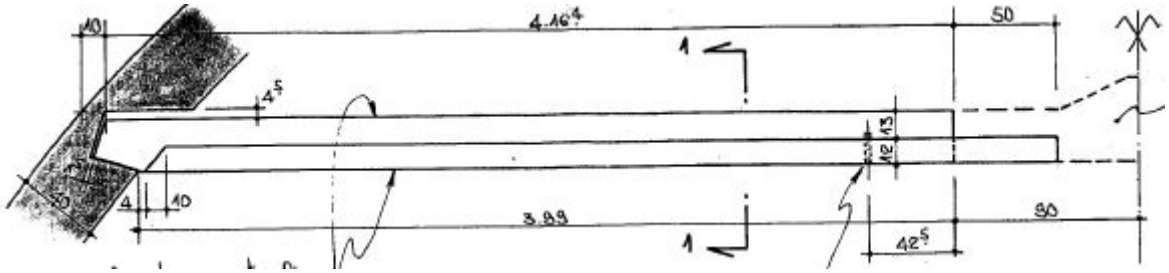

FIGURE 1.3 – Profil en long de la poutre en béton

Description géométrique profilé en T composé de 2 sections rectangulaires.

1. Première section rectangulaire ;
   - Epaisseur = 0,13 m,
   - Largeur = 0,4 m,
   - Section = 0,052 m^2,
   - Inertie = $\frac{Epaisseur^3 \times Largeur}{12} = 7,33 \times 10^{-}5\, m^4$;
2. Deuxième section rectangulaire ;
   - Epaisseur = 0,12 m,
   - Largeur = 0,9 m,
   - Section = 0,108 m^2,
   - Inertie = $1,3 \times 10^{-}4\, m^4$;
3. Section totale.

- Section totale = $Section_1 + Section_2 = 0,16\, m^2$,
- Inertie totale = $Inertie_1 + Inertie_2 = 2,0.10^{-4}\, m^4$,
- Longueur = $5\, m$;

Description mécanique poutres en béton armé. Puisque l'on considère un modèle de poutre de typer Euler-Bernoulli les propriétés mécaniques intéressantes sont la masse volumique ρ et le module d'élasticité E.

$\rightarrow$ $\rho = 2300\, kg.m^-3$;

$\rightarrow$ $E = 25GPa$.

Suspente acier béton

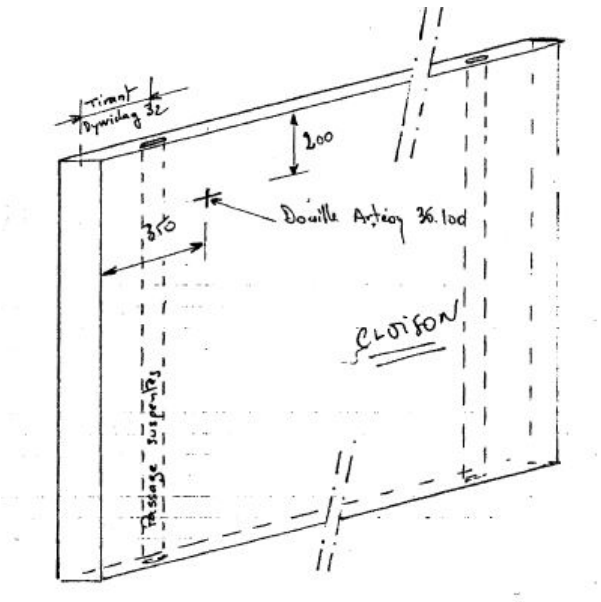

FIGURE 1.4 – Suspente mixte (acier, béton)

Description géométrique En réalité la suspente est un matériau à 2 composants. La suspente est une combinaison d'une barre cylindrique d'acier de 32mm de diamètre dans une paroi béton de 12cm d'épaisseur qui s'étend sur 5 mètres. La longueur de la suspente de 2.05 mètres. Ainsi pour modéliser cette poutre, on détermine un matériau homogène équivalent. Il nous faut donc déterminer l'épaisseur équivalente, le module d'Young équivalent et la masse volumique équivalente de ce matériau. On rappelle les différentes grandeurs propres à l'acier,

- Diamètre barre d'acier = $32\, mm$
- Section = $8,04.10^-4\, m^2$
- Inertie = $\frac{\pi.D^4}{64}$ = $5,15.10^-8\, m^4$
- Longueur = 2,05 m

Description mécanique acier Puisque on suppose une suspente cylindrique en acier, les caractéristiques mécaniques sont celles d'un acier classique de construction à savoir,

$\rightarrow$ $\rho =$ $7850kg.m^-3$

$\rightarrow E = 210GPa$

Détermination caractéristiques de la poutre équivalente

Les propriétés du matériau homogène équivalent sont notées E_{eq} (Module d'Young), e_{eq} (épaisseur), ρ_{eq}(masse volumique), A_{eq} et I_{eq}. Pour déterminer E_{eq} et I_{eq} on utilise les relations d'équilibres de rigidité en flexion et en compression suivantes,

$$\begin{aligned} E_{eq}\frac{He_{eq}^3}{12} &= E_bI_b + E_aI_a \\ E_{eq}He_{eq} &= e_bHE_b + S_aE_a \end{aligned}$$

Pour déterminer ρ_{eq} on utilise la formule, $\rho_{eq} = \frac{\rho_bS_b+\rho_aS_a}{S_{tot}}$. Pour les autres grandeurs on les détermine à partir des autres valeurs. Ainsi on obtient comme caractéristiques :

- $E_{eq} = 19,2GPa$
- $e_{eq} = 15,8cm$
- $H = 5m$
- $A_{eq} = e_{eq}H = 0,79m$
- $I_{eq} = \frac{e_{eq}^3H}{12} = 1,64.10^-3m^4$
- $\rho_{eq} = 1751,7kg.m^-3$

1.2 Résolution des équations

1.2.1 Présentation des équations

Soit une poutre de type Euler-Bernoulli ayant les caractéristiques géométriques et mécaniques suivantes,

Aire de la section	A
Longueur de la poutre	l
Inertie	I
Masse volumique	ρ
Module d'Young	E

Alors le comportement dynamique de cette poutre d'Euler Bernoulli est défini par les équations (chacune de ces équations caractérise un type de vibration, $\forall x \in [0, l]$)

- *Vibrations longitudinales*, ondes de compressions

$$\begin{aligned} &\text{Équation d'équilibre} && \frac{dN}{dx}(x) = \rho A\omega^2u(x) && (1) \\ &\text{Loi de comportement} && N(x) = -EA\frac{du}{dx}(x) && (2) \end{aligned}$$

- *Vibrations transversales*, ondes transversales

$$\begin{aligned} &\text{Équations d'équilibre} && \frac{dT}{dx}(x) = \rho A\omega^2v(x) && (3) \\ & && \frac{dM}{dx} = -T(x) && (4) \\ &\text{Loi de comportement} && M(x) = -EI\frac{d^2v}{dx^2}(x) && (5) \end{aligned}$$

Nous cherchons à déterminer l'expression de u et v en fonction du point d'abscisse x. u, v, N, T, M correspondent respectivement aux déplacements horizontaux, verticaux, à l'effort normal, tranchant et au moment fléchissant.

1.2.2 Détermination de u

Pour trouver u il suffit de remarquer l'égalité entre la dérivée de (2) par rapport à x et (1),

$$\frac{dN}{dx}(x) = -EA\frac{d^2u}{dx^2}(x) = \rho A\omega^2 u(x)$$

Soit,

$$\frac{d^2u}{dx^2}(x) + \frac{\rho\omega^2}{E}u(x) = 0$$

Il s'agit d'une équation de type oscillateur harmonique (lien avec l'équation du comportement d'un système masse-ressort $m\frac{d^2u}{dt^2} + ku = 0$).Ainsi, si l'on pose $\chi = \sqrt{\frac{\rho\omega^2}{E}}$ (le nombre d'onde de compression), alors u est de la forme

$$u(x) = u_1\cos(\chi x) + u_2\sin(\chi x) \quad \text{(N)}$$

Avec u_1 et u_2 constantes. Ces constantes peuvent être déterminées à partir des conditions aux limites de la poutre $u(0)$ et $u(l)$.

1.2.3 détermination de v

En dérivant (5) et en utilisant (4) on obtient une égalité et avec -(3). Ce qui permet d'obtenir l'équation différentielle linéaire homogène du quatrième ordre suivante,

$$\frac{d^4v}{dx^4} - \frac{\rho A\omega^2}{EI}v(x) = 0$$

On pose $\beta = \sqrt[4]{\frac{\rho A\omega^2}{EI}}$ (le nombre d'onde de flexion). Ainsi v est une combinaison linéaire de quatre fonctions indépendantes solutions de l'équation différentielle. Or les fonctions $\cos, \sin, \sinh, \cosh$ sont toutes solutions et indépendantes entre elles. Donc v est de la forme,

$$v(x) = v_1\cos(\beta x) + v_2\sin(\beta x) + v_3\cosh(\beta x) + v_4\sinh(\beta x) \quad \text{(N')}$$

Les conditions aux limites $v(0), v'(0), v(l), v'(l)$ permettent de déterminer les constantes v_1, v_2, v_3, v_4. Or dans l'hypothèse des petites déformations d'une poutre d'Euler-Bernoulli on a $\theta(x) \approx v'(x)$, avec θ rotation de la poutre par rapport à l'axe orthogonal du plan d'étude. Ainsi la connaissance de $(\theta(0), \theta(l), v(0), v(l))$ permet de déterminer les constantes de v.

1.2.4 Comparaison des longueurs d'ondes

Les longueurs d'ondes des ondes de compression et de flexion sont définies par,

- $\lambda_c = \frac{2\pi}{\chi l}$
- $\lambda_f = \frac{2\pi}{\beta l}$

Ainsi,

$$\frac{\lambda_c}{\lambda_f} = \frac{\beta}{\chi}$$

Après calculs on obtient

$$\frac{\lambda_c}{\lambda_f} = \frac{\sqrt[4]{12}}{\sqrt{2\pi}}\frac{\sqrt{\lambda_c}}{\sqrt{epaisseur}}$$

Dans le cas d'une poutre d'Euler-Bernoulli rectangulaire (Même si dans notre cas les poutres ne sont pas rectangulaires on pourrait très bien les modéliser par des poutres rectangulaires équivalentes de Bernoulli. Le résultat de ce calcul a donc bien un sens pour notre modèle.). De plus, on a fait l'hypothèse que l'épaisseur était faible devant la longueur d'onde donc $\frac{\lambda_c}{epaisseur} \gg 1$ et $\frac{\sqrt[4]{12}}{\sqrt{2\pi}}$ est de l'ordre de 1. On a donc $\lambda_c \gg \lambda_f$. Ainsi il existe des longueurs d'ondes où la poutre est en dynamique en compression et non en flexion.

1.3 Détermination des matrices élasto-dynamiques

1.3.1 Présentation

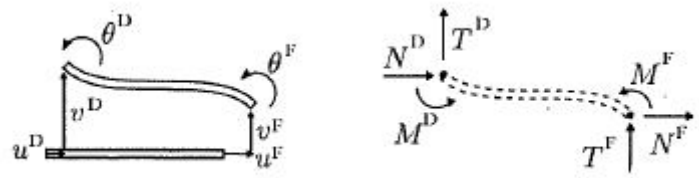

FIGURE 1.5 – Efforts et déplacements aux noeuds d'une poutre d'Euler-Bernoulli

Nous avons vu dans les sections précédentes que chaque u et v pouvaient être exprimées en fonction des conditions aux limites de l'élément poutre. De plus il est possible d'exprimer les forces N, T, M de la poutre en fonction des déplacements u et v. Ainsi il est également possible de déterminer les forces aux nœuds à partir des conditions aux limites. Il est même tout à fait possible d'exprimer ces forces sous la forme d'équations matricielles telles que,

$$\underline{f(0)} = \underline{\underline{HDD}}\,\underline{w(0)} + \underline{\underline{HDF}}\,\underline{w(l)} \quad (1.1)$$

$$\underline{f(l)} = \underline{\underline{HFD}}\,\underline{w(0)} + \underline{\underline{HFF}}\,\underline{w(l)} \quad (1.2)$$

Avec

- $\underline{f(l)} = \underline{f^f} = \begin{pmatrix} N(l) \\ T(l) \\ M(l) \end{pmatrix} = \begin{pmatrix} N^f \\ T^f \\ M^f \end{pmatrix}$
- $\underline{f(0)} = \underline{f^d} = \begin{pmatrix} N(0) \\ T(0) \\ M(0) \end{pmatrix} = \begin{pmatrix} N^d \\ T^d \\ M^d \end{pmatrix}$
- $\underline{w(l)} = \underline{w^f} = \begin{pmatrix} u(l) \\ v(l) \\ \theta(l) \end{pmatrix} = \begin{pmatrix} u^f \\ v^f \\ \theta^f \end{pmatrix}$
- $\underline{w(0)} = \underline{w^d} = \begin{pmatrix} N(0) \\ T(0) \\ M(0) \end{pmatrix} = \begin{pmatrix} N^d \\ T^d \\ M^d \end{pmatrix}$

L'objectif de cette partie est donc de déterminer les matrices $\underline{\underline{HDD}}, \underline{\underline{HDF}}, \underline{\underline{HFD}}, \underline{\underline{HFF}}$ appelées élasto-dynamiques. Grâce à elles tous les problèmes de dynamiques de structures en deux dimensions de type poutres de Bernoulli pourront être mis sous forme matricielle. Cela permettra un gain de temps, facilitera la détermination de l'équation aux valeurs propres et enfin facilitera la résolution de celle-ci.

Il sera simplement nécessaire de connaître les conditions aux limites de chaque élément de poutre (cela dépendra du modèle adopté), pour déterminer l'équation aux valeurs propres.

1.3.2 Mise en équation

Si l'on résout les problèmes de conditions aux limites évoquées en 1.2.2 et en 1.2.3 nous disposons du système de 6 équations à 6 inconnues linéaire suivant,

$$\begin{aligned}
u^d &= u_1\cos(0) + u_2\sin(0) \\
u^f &= u_1\cos(\chi l) + u_2\sin(\chi l) \\
v^d &= v_1\cos(0) + v_2\sin(0) + v_3\cosh(0) + v_4\sinh(0) \\
v^f &= v_1\cos(\beta l) + v_2\sin(\beta l) + v_3\cosh(\beta l) + v_4\sinh(\beta l) \\
\theta^d &= v_3\beta\sinh(0) - v_1\beta\sin(0) + v_4\beta\cosh(0) + v_2\beta\cos(0) \\
\theta^f &= v_3\beta\sinh(\beta l) - v_1\beta\sin(\beta l) + v_4\beta\cosh(\beta l) + v_2\beta\cos(\beta l)
\end{aligned}$$

Ce système peut donc se mettre sous la forme matricielle suivantes (on résout et on obtient les expressions de $u(x)$et $v(x)$ en réinjectant $\underline{C}$ dans N et N'.

$$\underline{\underline{A}}\,\underline{C} = \underline{w}^{df}$$

avec

- $\underline{\underline{A}} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ \cos(l\beta) & \sin(l\beta) & \cosh(l\beta) & \sinh(l\beta) & 0 & 0 \\ 0 & \beta & 0 & \beta & 0 & 0 \\ -\beta\sin(l\beta) & \beta\cos(l\beta) & \beta\sinh(l\beta) & \beta\cosh(l\beta) & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cos(l\chi) & \sin(l\chi) \end{pmatrix}$
- $\underline{C} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ u_1 \\ u_2 \end{pmatrix}$
- $\underline{w}^{df} = \begin{pmatrix} v_d \\ v_f \\ \theta_d \\ \theta_f \\ u_d \\ u_f \end{pmatrix}$

On peut également montrer qu'il existe une matrice $\underline{\underline{B}}$ (en réinjectant les déplacements u et v du vecteur $\underline{C}$ dans les équations de vibrations longitudinales (2) et transversales (4), (5) d'une poutre de Bernoulli) telle que,

$$\underline{\underline{B}}\,\underline{C} = \begin{pmatrix} N^d \\ T^d \\ M^d \\ N^f \\ T^f \\ M^f \end{pmatrix}$$

On cherche à obtenir les expressions des matrices élasto-dynamiques à partir des coefficients des matrices $\underline{\underline{A}}$ et $\underline{\underline{B}}$. L'idée est de remplacer le vecteur colonne des constantes $\underline{C}$, pour déterminer

un système matriciel liant les conditions aux limites en déplacements aux efforts en limites. Si $\det(\underline{\underline{A}}) \neq 0$. Alors $\underline{\underline{A}}$ est inversible et $\underline{\underline{C}} = \underline{\underline{A}}^{-1}\,\underline{w}^{df}$. On a donc

$$\underline{\underline{B}}\,\underline{\underline{A}}^{-1}\,\underline{w}^{df} = \begin{pmatrix} N^d \\ T^d \\ M^d \\ N^f \\ T^f \\ M^f \end{pmatrix}$$

Nous connaissons $\underline{\underline{A}}^{-1}$ et $\underline{\underline{B}}$. L'objectif est maintenant d'extraire les matrices élasto-dynamiques à partir de la matrice $\underline{\underline{D}} = \underline{\underline{B}}\,\underline{\underline{A}}^{-1}$.

1.3.3 Extraction des matrices

On remarque que si l'on crée une matrice composées des 3 premières lignes de $\underline{\underline{D}}$ et qu'on la multiplie par $\underline{w}^{df}$ on obtient la matrice colonne $\underline{f}^d$. Les colonnes 5,1 et 3 sont affectées aux conditions aux limites en 0 et les colonnes 6,2 et 4 sont affectées aux conditions aux limites en l. On remarque ainsi que,

$$\begin{pmatrix} \underline{\underline{HDD}} \\ \underline{\underline{HFD}} \end{pmatrix} = \left(\begin{array}{c|c|c} \underline{D_{5c}} & \underline{D_{1c}} & \underline{D_{3c}} \end{array} \right)$$

et

$$\begin{pmatrix} \underline{\underline{HDF}} \\ \underline{\underline{HFF}} \end{pmatrix} = \left(\begin{array}{c|c|c} \underline{D_{6c}} & \underline{D_{2c}} & \underline{D_{4c}} \end{array} \right)$$

On obtient ainsi les 4 matrices élasto-dynamiques suivantes :

$$\underline{\underline{HDD}} = \begin{pmatrix} \frac{AE\chi\cos(l\chi)}{\sin(l\chi)} & 0 & 0 \\ 0 & -\frac{EI\,\beta^3\,(\cos(l\beta)\sinh(l\beta)+\cosh(l\beta)\sin(l\beta))}{\cos(l\beta)\cosh(l\beta)-1} & -\frac{EI\,\beta^2\sin(l\beta)\sinh(l\beta)}{\cos(l\beta)\cosh(l\beta)-1} \\ 0 & -\frac{EI\,\beta^2\sin(l\beta)\sinh(l\beta)}{\cos(l\beta)\cosh(l\beta)-1} & \frac{EI\beta\,(\cos(l\beta)\sinh(l\beta)-\cosh(l\beta)\sin(l\beta))}{\cos(l\beta)\cosh(l\beta)-1} \end{pmatrix}$$

$$\underline{\underline{HFD}} = \begin{pmatrix} \frac{AE\chi}{\sin(l\chi)} & 0 & 0 \\ 0 & -\frac{EI\,\beta^3\,(\sinh(l\beta)+\sin(l\beta))}{\cos(l\beta)\cosh(l\beta)-1} & -\frac{EI\,\beta^2\,(\cosh(l\beta)-\cos(l\beta))}{\cos(l\beta)\cosh(l\beta)-1} \\ 0 & \frac{EI\,\beta^2\,(\cosh(l\beta)-\cos(l\beta))}{\cos(l\beta)\cosh(l\beta)-1} & \frac{EI\beta\,(\sinh(l\beta)-\sin(l\beta))}{\cos(l\beta)\cosh(l\beta)-1} \end{pmatrix}$$

$$\underline{\underline{HDF}} = \begin{pmatrix} -\frac{AE\chi}{\sin(l\chi)} & 0 & 0 \\ 0 & \frac{EI\,\beta^3\,(\sinh(l\beta)+\sin(l\beta))}{\cos(l\beta)\cosh(l\beta)-1} & -\frac{EI\,\beta^2\,(\cosh(l\beta)-\cos(l\beta))}{\cos(l\beta)\cosh(l\beta)-1} \\ 0 & \frac{EI\,\beta^2\,(\cosh(l\beta)-\cos(l\beta))}{\cos(l\beta)\cosh(l\beta)-1} & -\frac{EI\beta\,(\sinh(l\beta)-\sin(l\beta))}{\cos(l\beta)\cosh(l\beta)-1} \end{pmatrix}$$

$$\underline{\underline{HFF}} = \begin{pmatrix} -\frac{AE\chi\cos(l\chi)}{\sin(l\chi)} & 0 & 0 \\ 0 & \frac{EI\,\beta^3\,(\cos(l\beta)\sinh(l\beta)+\cosh(l\beta)\sin(l\beta))}{\cos(l\beta)\cosh(l\beta)-1} & -\frac{EI\,\beta^2\sin(l\beta)\sinh(l\beta)}{\cos(l\beta)\cosh(l\beta)-1} \\ 0 & -\frac{EI\,\beta^2\sin(l\beta)\sinh(l\beta)}{\cos(l\beta)\cosh(l\beta)-1} & -\frac{EI\beta\,(\cos(l\beta)\sinh(l\beta)-\cosh(l\beta)\sin(l\beta))}{\cos(l\beta)\cosh(l\beta)-1} \end{pmatrix}$$

Chapitre 2

Modélisation de la dalle de ventilation par modèle de poutre de flexion

2.1 Modèle appui simple et encastrement

2.1.1 Présentation du modèle à deux appuis et un encastrement

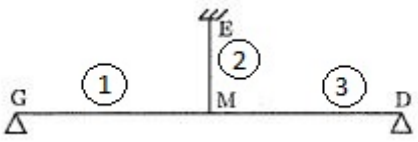

FIGURE 2.1 – modèle à 2 appuis et 1 encastrement

Le modèle est un ensemble de 3 poutres dans un plan. Ces poutres sont modélisées par des poutres d'Euler-Bernoulli ainsi il existe pour chacune d'elles un système matriciel comme celui détaillé dans le chapitre précédent. Puisque les poutres bétons et la suspente mixte ont des caractéristiques mécaniques et géométriques différentes, il est nécessaire de différencier les valeurs associées à chacune d'elles. Les grandeurs associées aux poutres bétons (respectivement suspente) seront indicées par b (respectivement a).

La poutre en béton à gauche de l'encastrement est numérotée 1, la poutre en béton à droite est numérotée 2 et la poutre en acier est numéroté 3. Le point G constitue le début de la poutre 1. Le point M constitue la fin de la poutre 1, le début de la 2 et de la 3. Le point D est la fin de la poutre 2. Le point E est la fin de la poutre 3. Ainsi on définit les quatre vecteurs déplacements aux nœuds $\underline{w}_G, \underline{w}_M, \underline{w}_D$ et $\underline{w}_E$. Chaque vecteur force issue des poutres se note $\underline{f}_i^{f,d}$ avec f, d la fin ou le début de la poutre et i le numéro de la poutre.

Les matrices élasto-dynamiques ont été définies dans le repère local de chaque poutre. Or dans ce cas ci le repère local de la poutre en acier diffère de ceux des poutres bétons. Lors de l'écriture des équilibres aux nœuds il faudra exprimer toutes les valeurs dans le même repère. Ainsi il est nécessaire d'introduire la matrice de rotation suivante,

$$\underline{\underline{R}}_\psi = \begin{pmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Celle ci permet de faire passer les valeurs d'un repère local à un autre en renseignant l'angle à faire faire au repère local pour qu'il soit dans le repère désiré. En l'occurrence le repère local de

la poutre en acier doit faire un angle de $-\frac{\pi}{2}$ pour être placé dans le repère global (qui correspond aux repères locaux des poutres 1 et 2). A noter que $\underline{w}_G, \underline{w}_M, \underline{w}_D$ sont exprimées dans le repère global et $\underline{w}_E$ dans le repère local de la poutre acier. Ainsi les équations matricielles dans le repère global de la poutre mixte sont :

$$\begin{aligned}
\underline{\underline{R}}_{-\frac{\pi}{2}}\,\underline{f}_3^f &= \underline{\underline{R}}_{-\frac{\pi}{2}}\,\underline{\underline{HFD}}_a\,\underline{\underline{R}}_{\frac{\pi}{2}}\,\underline{w}_M + \underline{\underline{R}}_{-\frac{\pi}{2}}\,\underline{\underline{HFF}}_a\,\underline{\underline{R}}_{\frac{\pi}{2}}\,\underline{\underline{R}}_{-\frac{\pi}{2}}\underline{w}_E \\
\underline{\underline{R}}_{-\frac{\pi}{2}}\,\underline{f}_3^d &= \underline{\underline{R}}_{-\frac{\pi}{2}}\,\underline{\underline{HDD}}_a\,\underline{\underline{R}}_{\frac{\pi}{2}}\,\underline{w}_M + \underline{\underline{R}}_{-\frac{\pi}{2}}\,\underline{\underline{HDF}}_a\,\underline{\underline{R}}_{\frac{\pi}{2}}\,\underline{\underline{R}}_{-\frac{\pi}{2}}\,\underline{w}_E
\end{aligned}$$

2.1.2 Équilibre aux nœuds et équation aux valeurs propres

Les conditions aux limites sont un encastrement en E, deux appuis simples en G, D et un encastrement des trois poutres en M. On peut donc déjà écrire que

$$\underline{w}_G = \begin{pmatrix} ug \\ 0 \\ \theta g \end{pmatrix} \quad \underline{f}_G = \begin{pmatrix} 0 \\ Tg \\ 0 \end{pmatrix}$$
$$\underline{w}_D = \begin{pmatrix} ud \\ 0 \\ \theta d \end{pmatrix} \quad \underline{f}_D = \begin{pmatrix} 0 \\ Td \\ 0 \end{pmatrix}$$
$$\underline{w}_E = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \underline{f}_E = \begin{pmatrix} Ne \\ Te \\ Me \end{pmatrix}$$

Nous disposons également d'un système de 4 équations matricielles traduisant chacun un équilibre des forces aux nœuds. Ainsi nous avons

Nœud	équilibre
G	$\underline{f}_G = \underline{f}_1^d$
D	$\underline{f}_D = \underline{f}_2^f$
E	$\underline{f}_E = \underline{f}_3^f$
M	$\underline{0} = \underline{f}_1^f - \underline{f}_2^d - \underline{\underline{R}}_{-\frac{\pi}{2}}\,\underline{f}_3^d$

Pour former l'équation aux valeurs propres il est nécessaire de récupérer toutes les équations qui sont égales à 0. Les équations égales à 0 sont les lignes 1 et 3 des systèmes matriciels en G et D, et les 3 lignes de l'équation matricielle en M. En remettant ce système de 7 équations sous forme matricielle en fonction de 7 déplacements inconnus on obtient

$$\underline{\underline{K}}_1\,\underline{U} \;=\; \underline{0}$$

avec

- $\underline{U} = \begin{pmatrix} ug \\ ud \\ um \\ vm \\ \theta g \\ \theta d \\ \theta m \end{pmatrix}$
- $\underline{\underline{K}}_1$ visible en annexe

Pour résoudre le problème aux valeurs propres il faut relever les fréquences pour lesquelles $\det(\underline{\underline{K}}_1) = 0$. Cependant ce déterminant est assez lourd et difficile à manipuler. Ainsi pour relever les différents modes nous allons réaliser une disjonction de cas pour lesquelles $\underline{Z}_1 = \underline{\underline{K}}_1\,\underline{U} = \underline{0}$. Celle-ci nous permettra d'obtenir différentes déformées associées à certaines fréquences.

2.1.3 Détermination des modes propres

Modes rigides

Nous disposons du vecteur colonne $\underline{Z}_1$ cependant celui-ci est assez étendu. On remarque cependant que les deux dernières équations de ce système matriciel valent

$$\begin{aligned} \frac{Ab\,Eb\,\chi b\ (ug\cos(lb\,\chi b) - um)}{\sin(lb\,\chi b)} &= 0 \\ -\frac{Ab\,Eb\,\chi b\ (ud\cos(lb\,\chi b) - um)}{\sin(lb\,\chi b)} &= 0 \end{aligned}$$

Ce sont les seules lignes de $\underline{Z}_1$ à ne dépendre que de des valeurs de u_D,u_G et de χ_b. Ainsi un premier mode est celui pour lequel $\chi_b\, l_b = \frac{\pi}{2}[\pi]$ et $u_m = 0$. Pour cette valeur de $\chi_b\, l_b$ les cinq autres lignes de $\underline{Z}_1$ ne peuvent être nulles ainsi cela impose $v_M = \theta_G = \theta_M = \theta_D = 0$. L'équilibre en M donne enfin $u_G = -u_D$. Les seules variables non nulles sont les déplacements longitudinaux.

Les fréquences propres associées sont donc $f_p = \frac{1}{2l_b}\sqrt{\frac{E_b}{\rho_b}}(\frac{1}{2} + p)\ \ \forall p \in \mathbb{N}$. Compte tenu des caractéristique mécaniques et géométriques on obtient pour les premières fréquences propres, les fréquences répertoriées dans le tableau ci-dessous.

Ordre	fréquence(Hz)
0	164,8
1	494,5
2	824,2

Puisque l'on échantillonne à 512 Hertz, nous pourrons seulement capter le premier mode rigide de la structure (théorème de Nyquist-Shannon $f_{max} \leq \frac{f_{echantillonnage}}{2}$).

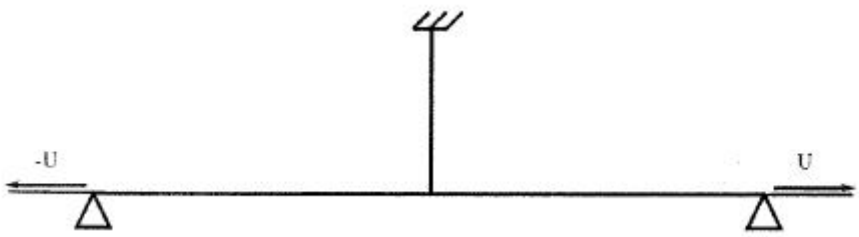

FIGURE 2.2 – Schéma des modes rigides

Modes transversaux

Si nous ne sommes pas dans le cas $\chi_b = \frac{\pi}{2l_b}[\pi]$ alors dans ce cas $u_m = u_g\cos(\chi_b l_b) = u_d\cos(\chi_b l_b)$. Dans ce cas là on que les deux premières lignes de $\underline{Z}_1$ (qui sont égales à zéro) ne dépendent que de $\theta_M, \theta_D, \theta_G, v_M$. Donc il est possible d'exprimer θ_D et θ_G en fonction de θ_M et v_M. Or les lignes 3,4,5 de $\underline{Z}_1$ dépendent de $\theta_M, \theta_D, \theta_G, v_M$ et u_m. Il est donc possible d'exprimer ces trois lignes seulement en fonction de θ_M, v_M et u_m. Ces trois lignes sont censées être égales à zéro. On obtient donc un système de 3 équations égales à 0 qui peut se mettre sous forme matricielle telle que

$$\underline{\underline{K}}_2 \begin{pmatrix} v_M \\ u_M \\ \theta_M \end{pmatrix} = \underline{0}$$

La matrice $\underline{\underline{K}}_2$ est diagonale par blocs ainsi la résolution de la résolution de $\det(\underline{\underline{K}}_2) = 0$ revient à trouver les valeurs de f pour lesquelles les déterminants des autres blocs sont égaux à zéro. Le

système est de cette sorte

$$\begin{pmatrix} k11 & 0 & 0 \\ 0 & k22 & k23 \\ 0 & k32 & k33 \end{pmatrix} . \begin{pmatrix} v_M \\ u_M \\ \theta_M \end{pmatrix} = \underline{0}$$

On remarque que les valeurs de f pour lesquelles $k11 = 0$ ne sont pas forcément les valeurs de f pour lesquelles $k22k33 - k23k32 = 0$. Ainsi il est important de faire la disjonction de 2 cas.

Déplacement de v_m

On a $k11 = 0 \Longrightarrow k22\,k33 - k23\,k32 \neq 0 \Longrightarrow u_m = \theta_m = 0 \Longrightarrow u_d = u_g = 0$. Précédemment on a exprimé (θ_d, θ_g) en fonction de (v_m, θ_m) si maintenant on a $\theta_m = 0$ alors on remarque que $\theta_d = -\theta_g = fonction(v_m)$. La fonction $k11(f)$ n'est pas simple à résoudre c'est pourquoi nous utilisons un schéma numérique pour trouver les valeurs f pour lesquelles $k11(f) = 0$. Sur le graphique (le graphique est tracé pour des fréquence entre 0 et 12 hertz cependant on répertorie toutes les fréquences comprises entre 0 et la moitié de la fréquence d'échantillonnage soit 256 Hertz) de $k11(f)$ il semble y avoir 1 points pour lequel k11 est nulle (l'autre est du à la discontinuité de la fonction). Après application d'un schéma numérique(méthode de la sécante), on obtient comme fréquences propres,

modes	fréquences (Hz)
1	11,44
2	37,04
3	77,18
4	131,72
5	200,29

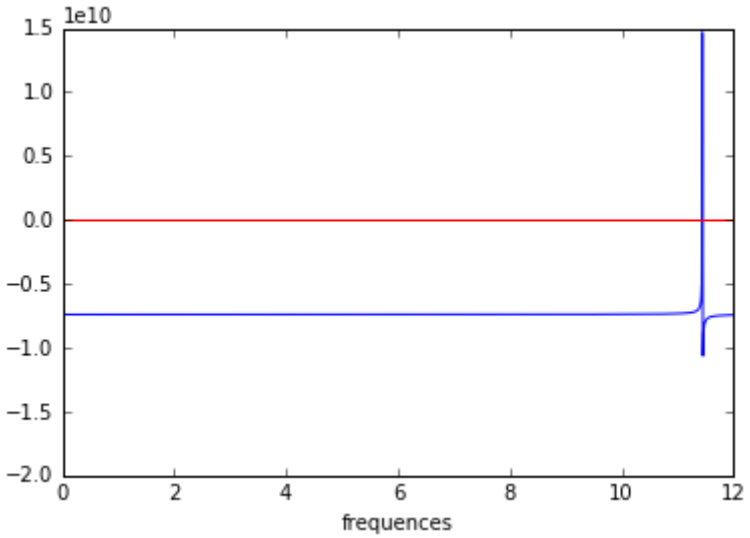

FIGURE 2.3 – Graphique de la fonction k11(f)

Déplacement de u_m et rotation de θ_m

On se place maintenant dans le cas où on a $k22k33 - k23k32 = 0$. Dans ce cas $v_m = 0$ et $(u_m, \theta_m) \neq (0,0) \Longrightarrow \theta_G = \theta_D = fonction(\theta_M)$. Et on rappelle que $u_m = u_g \cos(\chi_b l_b) = u_d \cos(\chi_b l_b)$. Ainsi si l'on trace $k22k33 - k23k32$ entre 0 et 256 hertz on observe peut être plus d'une dizaines de point pour lesquelles la fonction est nulle(certains sont dus à la discontinuité de la fonction, ils ne sont pas à prendre en compte). L'ensemble des fréquences perçues entre 0 et 256 hertz (par la méthode la sécante) sont,

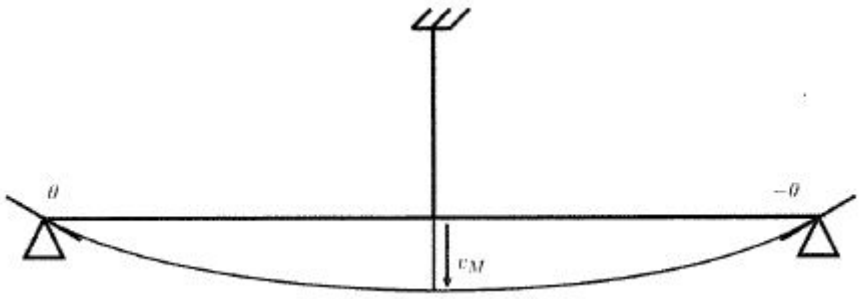

FIGURE 2.4 – Type de déplacements en v_m

modes	fréquences (Hz)
0	7,74
1	12,35
2	35,17
3	71,81
4	99,98
5	132,29
6	197,02

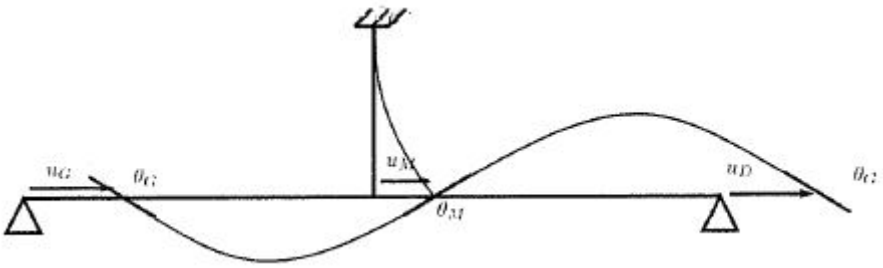

FIGURE 2.5 – Type de déplacements en u_m et rotation de θ_m

2.1.4 Comparaison avec un modèle numérique

A l'aide du logiciel ©Comsol nous avons réalisé une modélisation numérique de ce problème. L'idée est de comparer les résultats du modèle numérique avec le modèle analytique. Les résultats sont comparés dans le tableau ci-dessous.

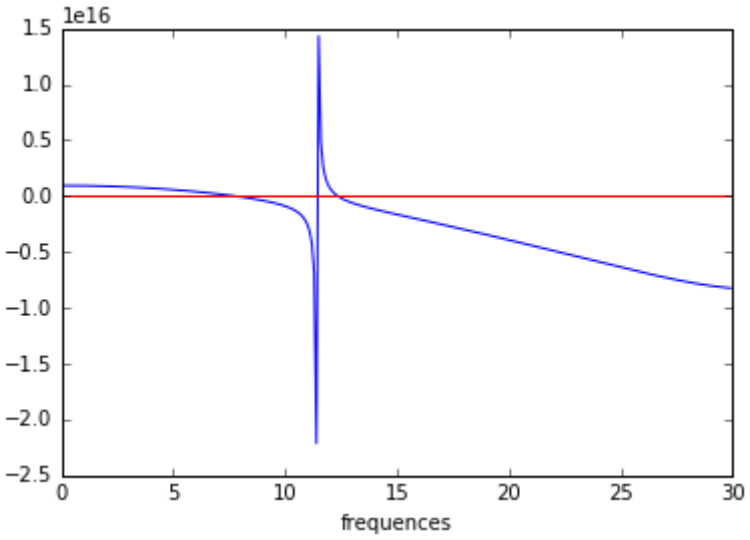

FIGURE 2.6 – Graphique de la fonction k22k33-k23k32

modes	cinématique	$f_{analytique}$ (Hz)	$f_{numerique}$ (Hz)	écart relatif(%)
0	u_m, θ_m	7,74	7,95	2,71
1	v_m	11,44	11,438	0,02
2	u_m, θ_m	12,35	12,557	1,68
3	u_m, θ_m	35,17	35,263	0,26
4	v_m	37,04	37,038	0,01
5	u_m, θ_m	71,81	72,256	0,62
6	v_m	77,18	77,185	0,01
7	u_m, θ_m	99,98	103,94	3,96
8	v_m	131,72	131,74	0,02
9	v_m	132,29	133,08	0,6
10	u_m	164,8	164,85	0,03
11	v_m	197,02	197,51	0,25
12	u_m, θ_m	200,29	200,38	0,04

L'écart relatif moyen est de 0,78 %. Compte tenu des faibles écarts entre les 2 modèles nous pouvons avoir une confiance certaine dans les résultats (relativement aux hypothèses formulées).

2.2 Modèle à trois encastrements

Précédemment nous avons expliqué pourquoi un modèle à 3 encastrements était plausible. On garde les mêmes grandeurs que pour la partie précédente.

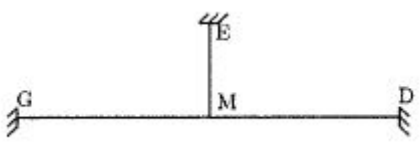

FIGURE 2.7 – Schéma du deuxième modèle

2.2.1 Conditions aux limites et équilibres nœuds

Nous avons 3 encastrement aux points G,D et E. Donc nous avons $\underline{w}_G = \underline{w}_E = \underline{w}_D = \underline{0}$. Ainsi nous n'avons que trois inconnues en déplacements à savoir u_M, v_M, θ_M. Or l'équilibre au point M s'écrit,

$$\underline{f}_1^f - \underline{f}_2^d - \underline{\underline{R}}_{-\frac{\pi}{2}}\, \underline{f}_3^d = \underline{0}$$

Cependant puisque les déplacements aux points G, D et E sont nulles on a

- $\underline{f}_1^f = \underline{\underline{HFF}}_b\, \underline{w}_M$
- $\underline{f}_2^d = \underline{\underline{HDD}}_b\, \underline{w}_M$
- $\underline{\underline{R}}_{-\frac{\pi}{2}}\, \underline{f}_3^d = \underline{\underline{R}}_{-\frac{\pi}{2}}\, \underline{\underline{HDD}}_a\, \underline{\underline{R}}_{\frac{\pi}{2}}\, \underline{w}_M$

Ainsi si l'on pose $\underline{\underline{K}}_3 = \underline{\underline{HFF}}_b - \underline{\underline{HDD}}_b - \underline{\underline{R}}_{-\frac{\pi}{2}}\, \underline{\underline{HDD}}_a\, \underline{\underline{R}}_{\frac{\pi}{2}}$ on a $\underline{\underline{K}}_3\, \underline{w}_M = \underline{0}$

Cependant on remarque que la matrice $\underline{\underline{K}}_3$ à une forme particulière. En se rappelant que cette matrice est en fait un système d'équations linéaires formé des lignes du produit $\underline{\underline{K}}_3\, \underline{w}_M$ on peut en fait transformer la représentation de ce système sous la forme

$$\underline{\underline{K}}_4 \begin{pmatrix} vm \\ um \\ \theta m \end{pmatrix} = \underline{0}$$

avec $\underline{\underline{K}}_4$ une matrice diagonale par blocs de la même sorte que $\underline{\underline{K}}_2$. Ainsi comme précédemment il y a un découplage entre les cinématiques par rapport à v_M et celles par rapport à (u_m, θ_M).

2.2.2 Cinématique par rapport à v_M

On a une équation de la forme $g(f).v_M = 0$. On recherche donc les valeurs fp tel que $g(fp) = 0$. Ainsi après avoir tracé le graphique de cette fonction on observe plusieurs points d'intersections avec l'axe des abscisses. Après utilisation d'un schéma numérique on trouve comme valeurs,

modes	fréquences (Hz)
1	16,6
2	45,71
3	89,48
4	147,57
5	219,51

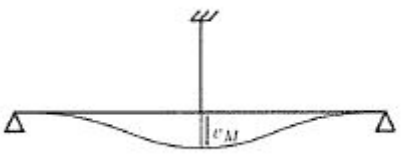

FIGURE 2.8 – Type de déplacements

2.2.3 Cinématique par rapport à u_M, θ_M

On s'intéresse ici à l'équation matricielle suivante

$$\begin{pmatrix} h_1(f) & h_2(f) \\ h_3(f) & h_4(f) \end{pmatrix} \cdot \begin{pmatrix} u_M \\ \theta_M \end{pmatrix}$$

En particulier aux valeurs fp pour lesquelles $h_1(f_p)h_4(f_p) - h_2(f_p)h_3(f_p) = 0$. On trace donc sa fonction est on obtient les fréquences propres suivantes,

modes	fréquences (Hz)
0	11,48
1	26,77
2	37,2
3	73,52
4	77,88
5	132,19
6	145,24
7	164,91
8	221,55
9	240,09

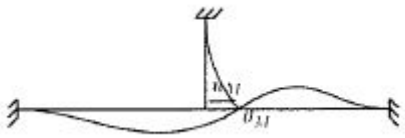

FIGURE 2.9 – Type de déplacements

2.3 Limites du modèle et conclusion partielle

Un modèle analytique de la structure permet d'avoir une vision sur le comportement la structure. Ce modèle nous permettra d'aborder les résultats trouvés de manière opérationnelle avec un regard critique. En effet, l'observation des résultats sans une certaine référence (modèle analytique) peut s'avérer infructueuse. Cependant chaque modèle a ses limites qu'il convient de préciser.

2.3.1 Critique 1

Le modèle de poutre utilisé permet de capter la quasi-totalité des cinématiques en plan. Cependant, une poutre est un élément en trois dimensions il peut donc exister certaines vibrations en deux dimensions. Ces vibrations ne sont pas prises en compte par notre modèle. Car un modèle de poutre de Bernoulli (Tout comme un modèle de Timoshenko) considère des vibrations seulement en deux dimensions. Un modèle de poutre de Bernoulli a été privilégié à un modèle de Timoshenko car il traduit mieux le comportement des poutres élancées et qu'il est moins difficile à utiliser. Enfin notre modèle reste pertinent car lors de la relève des modes propres réels de la structure, les modes les plus simples à récupérer sont ceux issus de notre modèle. Il est donc pertinent d'utiliser notre modèle dans un souci d'efficacité.

2.3.2 Critique 2

La deuxième critique concerne la modélisation de la suspente par une poutre constitué d'un matériau homogène équivalent. Ce type de modélisation est valable en cas d'éléments composites qui n'interagissent pas entre eux. C'est généralement le cas entre l'acier et le béton. Cependant il se peut que l'acier s'oxyde. De plus le module du béton diminue avec le temps ainsi le matériau homogène équivalent aurait des propriétés qui pourrait changer en fonction du temps et des différentes réactions qui pourraient subvenir.

Enfin la suspente est assez étendue hors du plan et nous la modélisons par une poutre élancée, alors que sa forme est plus proche d'une plaque. Cela peut changer la valeur du module d'Young et à fortiori des fréquences propres calculées.

2.3.3 Critique 3

Le béton à un module qui diminue et nous avons pris un module pour du béton jeune. Or la dalle fut mise en place 30 ans avant la date des premières mesures. Sachant que les fréquences déterminées avec notre modèle dépendent du module du béton il se peut que les fréquences ne soient pas exactement les mêmes.

2.3.4 Conclusion partielle

Même si diverses critiques peuvent être formulées quant au modèle utilisé, notre modèle permet d'avoir une bonne estimation des différentes fréquences propres de la structure et nous sommes également sûrs des résultats obtenus analytiquement (grâce à la comparaison avec un modèle numérique. Les deux modèles ont des résultats très proches). A l'aide des mesures qui seront effectuées par la suite nous pourrons trancher sur les deux modèles proposés (trois encastrements ou un encastrement et deux appuis simples).

Deuxième partie

Analyse Modale Opérationnelle

Cette partie est dédiée aux méthodes de traitement du signal utilisées dans le génie civil. Elle s'intéresse aux vibrations des structures de génie civil. Or plusieurs hypothèses sont faites dans le cadre de la dynamique en génie civil, qu'il convient de rappeler. Cette liste non exhaustive rappelle les principales hypothèses :

- La première est le comportement linéaire des structures de génie civil. Cette hypothèse permet de s'assurer de l'existence d'une fonction de transfert caractérisant la structure. Dans le cadre de notre étude cette hypothèse est tout à fait réaliste car nous avons à faire à de petits déplacements.
- La quantité suffisante d'information est la seconde hypothèse. Étant donné que nous avons réalisé les mesures pendant 30 minutes cette hypothèse semble également raisonnable.
- Durant les mesures on suppose que les paramètres physiques de la structure ne changent pas. Ce qui est réaliste car la structure n'est pas soumise à des sollicitations pouvant l'endommager durant le test.
- Les signaux relevés sont supposés être stationnaire. C'est à dire que les propriétés statistiques des signaux sont indépendants du temps. Étant donné que l'excitation est due au trafic routier et que celui-ci est quasiment continu durant toute la mesure on peut raisonnablement considérer que le signal relevé sera stationnaire.
- Enfin on suppose que le signal (qui peut être vu comme un processus aléatoire) est un processus ergodique. C'est à dire que les caractéristiques du signal peuvent être déterminée par une seule réalisation (si cette réalisation contient suffisamment d'informations $\Rightarrow$ seconde hypothèse). Sans cette hypothèse l'analyse des signaux est impossible.

Chapitre 3

Revue des principales techniques d'analyse modale opérationnelle

Plusieurs méthodes d'analyse modale existent. Deux méthodes en particulier sont prépondérantes. Celles-ci sont l'analyse modale opérationnelle (OMA) et l'analyse modale expérimentale (EMA). L'EMA est très répandue notamment dans l'industrie. Elle estime les paramètres modaux (fréquence, amortissement et déformées) en fonction de l'excitation et de la réponse du système. Cette méthode a connu beaucoup d'avancées ces dernières années avec des algorithmes de traitement du signal à partir de modèles SISO (une entrée, une sortie), SIMO (une entrée, plusieurs sorties) et MIMO (plusieurs entrées, plusieurs sorties).

Cependant, l'EMA a des limites. La première est qu'elle nécessite une excitation artificielle (indispensable pour connaitre le signal d'entrée). Or dans le cas d'infrastructures de tailles importantes, il est difficile de mesurer un signal de sortie à partir d'une faible excitation et en l'occurrence les excitations artificielles sont généralement des excitations de faible amplitude. De plus, l'EMA est souvent réalisée en laboratoire donc dans des conditions autres que son exposition traditionnelle. Lorsque ces mesures sont réalisées sur place, sur des structures de génie civil, il est souvent nécessaire de devoir mettre en place des mesures contraignantes (arrêt du trafic, conditions météorologiques adéquates, utilisation de balourds). En effet le trafic routier ou le vent pourrait perturber l'excitation artificielle (excitations utilisées dans l'OMA).

C'est pourquoi à partir des années 1990, l'analyse modale opérationnelle s'est largement développée dans le génie civil. Elle s'applique dans beaucoup de secteurs de celui-ci, des plates-formes offshores, aux bâtiments en passant par les ponts, les gratte-ciels, etc. Contrairement à l'EMA, l'OMA ne mesure que le signal de sortie de la structure, dans des conditions normales d'utilisation. Elle pose donc des hypothèses sur le signal d'entrée. L'OMA est peu coûteuse et rapide à réaliser. Elle permet d'obtenir des caractéristiques dynamiques macroscopiques de la structure. Enfin cette méthode ne permet pas seulement de caractériser la structure, elle permet également de contrôler les vibrations des structures et de vérifier leur endommagement. C'est pourquoi l'OMA est très attractive et qu'elle s'étend dans d'autres domaines que le génie civil (aérospatial, génie mécanique). L'OMA a également des limites comparée à l'EMA. En effet, les tests avec les vibrations ambiantes peuvent ne pas couvrir l'ensemble de fréquences qui nous intéresse. En cas de vibrations forcées l'ensemble des fréquences intéressantes sont couvertes. En particulier en cas de modes en compression longitudinales qui ont des fréquences propres dans les hautes fréquences.

Il existe plusieurs types d'OMA. Ces différents types sont détaillés dans ce chapitre. La plupart de ces méthodes supposent que le bruit d'entrée est un bruit blanc. Cette hypothèse n'est pas tout à fait réaliste. L'entrée peut en réalité être considérée comme étant la sortie d'un filtre linéaire soumis à un bruit blanc. Ce qui signifie que l'identification des modes n'est pas seulement associé à la structure mais également au système source-structure qui a produit l'excitation réelle. En pratique il est possible de séparer les paramètres modaux associés au

système de la structure des paramètres modaux du système source-structure.

3.1 OMA dans le domaine temporel

3.1.1 Natural Excitation Technique (Next)

Au début des années 90 cette technique fût proposée pour l'identification modale uniquement à partir des mesures de sorties. La méthode NExT utilise la fonction de corrélation pour l'analyse d'une réponse aléatoire de la structure à une excitation naturelle. La fonction de corrélation discrète croisée $R_{X,Y}$ peut être estimée de cette façon :

$$R_{X,Y}(k\Delta t) = \frac{1}{L-k}\sum_{n=1}^{L-k} x_n y_{n+k}$$

avec

- $X = [x_1, ..., x_L]$ et $Y = [y_1, ..., y_L]$ deux séries de données recueillies.
- Δt le pas de temps d'acquisition du signal
- L le nombre total de points acquis
- si $X = Y$ on parle alors d'auto-corrélation. Sinon on parle d'inter-corrélation

Selon la méthode NExt les fonctions de corrélations peuvent être découpées en une somme de sinusoïdes décroissantes. Ces sinusoïdes décroissantes possèdent une fréquence naturelle, un coefficient d'amortissement et une déformée modale. Ces fréquences naturelles, coefficients d'amortissement et déformées modales sont les mêmes que les réponses impulsionnelles du système (Le développement théorique est réalisé dans l'article *'The natural excitation technique (NExT) for modal parameter extraction from operating structures'* de G.H James et Thomas G.Carne publié en Janvier 1995). Ainsi en déterminant les paramètres modaux de la fonction de corrélation on obtient les paramètres modaux du système. Les fonctions de corrélations peuvent par la suite être employées comme des fonctions de réponses impulsionnelles pour estimer les paramètres modaux. On remarque que les principales procédures d'identifications modales dans le domaine temporel développée en EMA peuvent être utilisées en OMA. En effet la méthode NExT est un cas particulier des méthodes d'experimental modal analysis où l'on considère que le bruit d'entrée est un bruit blanc.

Ainsi si l'on récupère plusieurs séries de données on peut les croiser entre elles pour obtenir les fonction de corrélation croisées. Les fonctions de corrélations peuvent être estimées à partir de l'estimateur précédent ou à partir du corrélogramme (estimateur de la densité spectrale de puissance pour l'auto-corrélation ou de l'inter-spectre pour estimée l'inter-corrélation, ces notions seront détaillées en 1.2). Ces fonctions de corrélations peuvent ensuite être utilisée comme des fonctions de réponses impulsionnelles dans le domaine temporel. Or, il existe des algorithmes qui permettent d'extraire les paramètres modaux des réponses impulsionnelles. Ainsi l'identification modale de la structure peut se faire par le biais de d'algorithmes tels que l'exponentielle complexe poly-reference (PRCE), l'algorithme de réalisation du sytème propre (ERA) et le domaine temporel étendu d'Ibrahim (EITD) appliquées aux fonctions de corrélations.

3.1.2 Procédures auto-régressives(AR) et auto-régressives à moyennes mobiles (ARMA)

En traitement du signal deux catégories de signaux sont actuellement employés pour estimer la densité spectrale de puissance. Les traitements dits 'classiques' qui reposent sur la transformée de Fourier du signal ou sur la transformée de Fourier de sa fonction d'auto-corrélation. L'autre

famille de traitements, dits 'modernes', s'appuie sur une modélisation paramétrique de la densité spectrale de puissance (Fabrice HEITZ, septembre 2014, *Traitement des signaux aléatoires*). Cette sous-section s'intéresse en particulier aux procédures de types auto-régressives et auto-régressives à moyennes mobiles.

Procédure auto-régressive (AR)

Selon cette méthode tous les signaux à un instant $n = k\Delta t$ (signal discrétisé) peuvent être exprimés comme une combinaison linéaire des p (p étant appelé l'ordre du modèle) termes précédent, soit

$$\forall x_n \in [p, N], \quad x_n = -\sum_{k=1}^{p} a_k x_{n-k}$$

avec

- a_k les paramètres du modèle AR
- p l'ordre du modèle
- N le nombre de termes du signal $X = [x_1, ..., x_N]$

En réalité le x_n est un prédicteur de la valeur réelle x_n (on notera le prédicteur d'ordre p $x_{p,n}$). Ainsi lorsque l'on souhaite modéliser un processus (notre signal étant une suite de variables aléatoires) par un modèle AR d'ordre p on cherche à trouver les meilleurs a_k tels que : $\rho = E(|x_{p,n} - x_n|^2)$ (E est l'espérance mathématique, içi il s'agit de la moyenne des écarts au carré) soit minimal. On appelle ρ l'erreur de prédiction linéaire.

Les algorithmes principaux qui permettent de retrouver ces paramètres a_k sont les suivants. Le premier est l'algorithme de Levinson Durbin appliqué aux système matriciel des équations de Yule-Walker (Les équations de Yule-Walker sont des équations sur les a_k qui permettent d'obtenir l'erreur de prédiction linéaire minimale). C'est un algorithme récursif en ordre, c'est-à-dire qu'il calcul les paramètres a_k d'un ordre p en fonction des paramètres a_k d'un ordre p moins un. Enfin on peut utiliser un algorithme des moindres carrés qui minimisent la puissance de l'erreur de prédiction linéaire. Les avantages et inconvénients des méthodes de détermination des a_k sont détaillés par la suite.

La grande difficulté de cette méthode de traitement du signal est la détermination de l'ordre du modèle. En effet un ordre trop faible ou trop élevé ne traduirait pas correctement le comportement de la structure (un ordre faible lisse la densité spectrale de puissance, un trop grand entraîne l'apparition de modes secondaires). La détermination peut se baser sur plusieurs critères de détermination. Trois de ces critères sont récurrents : le critère d'informations de Akaike, le critère d'information de Bayes et la fonction d'auto-corrélation partielle. En réalité selon Olivier BESSON, *'Analyse spectrale paramétrique'*, il est difficile d'estimer l'ordre du modèle par ces critères.

Dans le cas d'un processus AR la valeur de la densité spectrale de puissance pour une fréquence f est :

$$S_x(f) = \frac{\sigma^2}{|1 + \sum_{k=1}^{p} a_k e^{-j2\pi kf}|^2}$$

Ainsi pour retrouver les fréquences propres d'un système un modèle AR suffit (σ^2 est la variance du bruit d'entrée, puisque nous n'avons pas d'information sur le bruit d'entrée nous déterminons la densité spectrale de puissance à un coefficient multiplicatif près). Cependant les procédures ARMA permettent d'obtenir des densités spectrales de puissances plus 'lisibles'.

Procédure auto-régressive à moyenne mobile (ARMA)

Selon cette méthode tous les signaux à un instant $n = k\Delta t$ peuvent être exprimés comme une combinaison linéaire des p (p étant appelé l'ordre auto-régressif du modèle) termes précédent du signal de sortie et $q + 1$ termes du signal d'entrée du système, soit

$$x_n = \sum_{k=1}^{p} -a_k x_{n-k} + \sum_{k=1}^{q} b_k u_{n-k}$$

avec

- a_k les paramètres du modèle AR
- p l'ordre du modèle auto-régressif
- q l'ordre des paramètres de moyenne mobile
- u le signal d'entrée du système

Il est démontrée que pour un bruit blanc d'entrée (ce qui est le cas en analyse modale opérationnelle) la densité spectrale de puissance peut être exprimée par,

$$S_x(f) = \sigma^2 \frac{|1 + \sum_{k=1}^{q} b_k e^{-j2\pi kf}|^2}{|1 + \sum_{k=1}^{p} a_k e^{-j2\pi kf}|^2}$$

Ainsi il est tout à fait possible d'obtenir une estimation de la densité spectrale de puissance seulement à partir des paramètres ARMA à un coefficient multiplicatif (σ^2 la variance du bruit d'entrée) près.

De la même manière que pour une procédure AR, une procédure ARMA est soumise au choix de l'ordre du modèle. Les mêmes difficultés de choix existent donc pour un modèle ARMA.

Détermination des fréquences propres et amortissements modaux grâce aux paramètres AR

Admettons que l'on modélise notre signal par une procédure AR d'ordre p noté (AR(p)) alors la densité spectrale de puissance peut être exprimée de la façon suivante.

$$S_x(f) = \frac{\sigma^2}{|1 + \sum_{k=1}^{p} a_k e^{-j2\pi \frac{kf}{f_e}}|^2}, \text{ pour } f \in [0; \frac{f_e}{2}]$$

Si on pose $z = e^{-j2\pi f}$ alors $z^k = e^{-j2\pi kf}$ et donc la recherche des pics de la fonction $S_x(f)$ revient à une recherche de zéros du polynôme $\mathcal{A}(z) = 1 + a_1 z + \cdots + a_p z^p$. Puisque les a_k sont des coefficients réels alors les racines complexes de z seront des paires de racines composées d'une racine complexe et de son conjugué. En outre, d'après la thèse Clotaire Michel, les racines peuvent être mises sous la forme $p_{1,2} = e^{(-\xi 2\pi f_p \pm j2\pi f_p)\Delta t}$. Donc après avoir déterminé les racines de $\mathcal{A}$ on peut obtenir les fréquences de telle sorte que

$$f_p = \frac{Arg(p_{1,2})}{2\pi\Delta t}$$

et l'amortissement associé,

$$\xi_p = \frac{ln(||p_{1,2}||)}{2\pi f_p \Delta t}$$

avec $p_{1,2}$ une paire de racines complexes conjuguées.

Critères d'aide à la détermination de l'ordre

On choisit les ordres p qui minimisent ces critères.

- Le critère d'information d'Akaike est défini par la formule (k est l'ordre du modèle, N est le nombre de points du signal, $\widehat{\rho_k}$ est l'erreur de prédiction linéaire) :

$$AIC(k) \;=\; N\ln(\widehat{\rho_k}) \;+\; 2k$$

- Le critère d'information de Bayes est défini par la formule :

$$BIC(k) \;=\; -2\ln(\widehat{\rho_k}) \;+k\ln(N)$$

- le principe de la fonction d'auto-corrélation partielle est expliqué par la suite.

3.1.3 Méthode du décrément aléatoire

Elle est proposée pour la première fois par H. Cole avant 1970, alors ingénieur à la NASA. Cette méthode utilise un calcul de probabilité conditionnelle. Elle est généralement utilisée pour déterminer les coefficients d'amortissements et pour détecter les défauts mécaniques. L'objectif du décrément aléatoire est de construire des fonctions, appelées fonctions de décrément, à partir du signal de sortie afin d'en extraire les paramètres modaux d'une structure. Elle a l'avantage d'être rapide et peu coûteuse en calcul. Le décrément aléatoire est en fait un estimateur de la fonction de corrélation. Cette fonction de corrélation est propre à des signaux aléatoires stationnaires. A partir de l'estimation de la fonction de corrélation il est possible d'extraire des paramètres modaux comme on l'a vu précédemment dans la méthode NExT. Cet estimateur est beaucoup moins coûteux en temps de calcul par rapport à l'estimation classique de la fonction d'auto-corrélation.

3.2 OMA dans le domaine fréquentiel

3.2.1 Le 'Peak Picking' ou relevé de pics

Ces méthodes sont basées sur la formule de la densité spectrale de puissance dans le cas d'un processus stochastique (dans notre cas le processus stochastique est le signal de sortie récupéré). Il convient donc d'estimer la densité spectrale de puissance. Parmi les estimations de la densité spectrale de puissance il existe deux méthode classiques, le périodogramme et le corrélogramme. La méthode du relevé de pics consiste à récupérer les pics de la densité spectrale de puissance. Ces pics sont censés être situées aux fréquences propres de la structure. Cependant en analysant les différentes méthodes d'estimations de la densité spectrale de puissance plusieurs remarques peuvent être formulées.

Le périodogramme estime la densité spectrale de puissance en calculant le module au carré (transposée de la matrice de réponse fréquentielle avec le conjugué de la matrice de réponse fréquentielle) de la transformée de Fourier du signal x(t). Ainsi pour un signal discret $\underline{x}$ composé de N points échantillonné à une fréquence $f_e = \frac{1}{T_e}$ on obtient comme estimé,

$$S_{e,N}(f) = \frac{1}{N}|\sum_{k=0}^{N-1} x_k e^{-j2\pi kT_e f}|^2$$

Le périodogramme est un estimateur biaisé, c'est-à-dire que la moyenne de l'estimateur n'est pas égale à la moyenne du signal (mais pour un nombre de points important il peut être considéré sans biais). Par contre il est démontré que sa variance ne diminue pas en fonction du nombre de points et que cette variance de l'estimateur est maximale pour les pics. Ainsi aux endroits

où nous souhaitons récupérer des valeurs la variance est la plus élevée, les valeurs que nous récupérons peuvent donc être éloignées des valeurs réelles (Fabrice HEITZ, septembre 2014, *Traitement des signaux aléatoires*). L'estimation étant trop éloignée de la réalité.

Le corrélogramme lui estime la densité spectrale de puissance à partir de la transformée de Fourier d'une estimation de la fonction d'auto-corrélation de x(t) (comme celle vu dans la méthode NExT). C'est le théorème de Wiener-Kinchine qui démontre que la transformée de Fourier de la fonction d'auto-corrélation est égale à la densité spectrale de puissance. Ainsi on a,

$$S_{e,N}(f) = \sum_{k=-\infty}^{+\infty} R_{X,X} e^{-j2\pi k T_e f}$$

Dans le cas de signaux réels le corrélogramme et le périodogramme sont égaux (ce qui est le cas du signal relevé). Ainsi le corrélogramme est également un mauvais estimateur spectral.

Ainsi pour éviter d'avoir des trop grands écarts par rapport aux valeurs 'réelles' des fréquences propres il est parfois nécessaire de lisser ces estimateurs. Pour cela il faut réaliser un fenêtrage du signal. Le fenêtrage du signal consiste à multiplier la fonction d'auto-corrélation dans le cas du corrélogramme par une fenêtre d'apodisation ou d'observation $w_M(t)$. Plusieurs types de fenêtres existent comme par exemple la fenêtre rectangulaire, triangulaire, de Hann, de Hamming... Utiliser une fenêtre d'observation revient à observer le signal sur une durée finie qui correspond à la largeur de la fenêtre utiliser. M correspond à la largeur du fenêtrage (qui peut être exprimé en nombre de points en cas de signal discret). Le lissage permet de réduire significativement la variance. Même s'il augmente le biais, il reste un estimateur fiable. Pour réduire la variance il est nécessaire que M soit suffisamment important mais qu'il reste toutefois très faible devant le nombre de point N du signal discret (Soit $N \gg M$).

Le 'Peak picking' est une bonne première estimation des fréquences propres d'un signal périodique stationnaire. L'abscisse des pics donne une première approximation des fréquences propres, leur ordonné donne une indication sur l'amplitude de ce mode et la largeur permet d'obtenir une approximation de l'amortissement modal. Cependant l'amortissement estimé est souvent de mauvaise qualité. Cependant selon la thèse de C. Michel 2007, le relevé de pics 'permet seulement d'avoir une estimation biaisée des modes car il ne les décompose pas'. C'est-à-dire que la déformée obtenue par cette méthode est une composition des modes ayant de l'énergie à cette fréquence. On peut bien sûr supposer que les modes obtenus sur une direction sont indépendants des modes dans les autres directions, en réalité ce n'est pas forcément le cas (comme expliqué dans les limites de la partie 1).

3.2.2 La Frequency Domain Decomposition (FDD)

La FDD est une amélioration de la méthode des pointés de pics. Elle fût présentée en 2000 par Brincker et al. dans l'article *" Modal identification from ambient responses using frequency domain decomposition"*. Elle est décomposée en trois étapes : l'estimation de la matrice de densités spectrales de puissances croisées, la décomposition en valeurs singulières de celle-ci, le relevé de pics de valeurs singulières calculées.

Admettons que l'on mesure n signaux de taille N, la matrice de densité spectrale de puissance croisée est de taille $n \times n \times N$. Il s'agit donc d'une matrice à trois dimensions. Dans cette matrice sont disposées à la ligne i et en colonne j les transformées de Fourier des fonctions d'inter-corrélations entre signaux de taille N. Ainsi à la profondeur n la matrice de densité spectrale de puissance est :

$$\underline{\underline{G}}(k\Delta f) = \begin{pmatrix} S_{e,N,X_1,X_1}(k\Delta f) & S_{e,N,X_1,X_2}(k\Delta f) & \cdots & S_{e,N,X_1,X_n}(k\Delta f) \\ S_{e,N,X_2,X_1}(k\Delta f) & S_{e,N,X_2,X_2}(k\Delta f) & \cdots & S_{e,N,X_2,X_n}(k\Delta f) \\ \vdots & \vdots & \ddots & \vdots \\ S_{e,N,X_n,X_1}(k\Delta f) & S_{e,N,X_n,X_2}(k\Delta f) & \cdots & S_{e,N,X_n,X_n}(k\Delta f) \end{pmatrix}$$

avec

- $X_1, \cdots, X_n$ les différents signaux enregistrés (n listes de N termes)
- $S_{e,N,X_i,X_j}(k\Delta f) = \sum_{n=-\infty}^{+\infty} R_{X_i,X_j} e^{-j2\pi n T_e k\Delta f}$ si l'inter-spectre est estimé en utilisant la fonction d'inter-corrélation (approche du corrélogramme)
- $S_{e,N,X_i,X_j}(k\Delta f) = TF(X_i)TF(X_j^*)$ si l'inter-spectre est estimé par transformée de Fourier des signaux (approche du périodogramme)
- Δf la résolution fréquentielle

La deuxième étape consiste à effectuer une décomposition en valeurs singulières des matrices $\underline{\underline{G}}(k\Delta f) \forall\ k\ \in [0, N]$. C'est à dire à exprimer $\underline{\underline{G}}(k\Delta f)$ tel que $\underline{\underline{G}}(k\Delta f) = \underline{\underline{U_k}}\ \underline{\underline{S_k}}\ \underline{\underline{U_k^H}}$. La matrice $\underline{\underline{U_k^H}}$ est la matrice adjointe (transposée de la matrice conjugué) de $\underline{\underline{U_k}}$. Les diagonales de $\underline{\underline{S_k}}$ sont les valeurs singulières. En termes d'interprétation statistique, les valeurs des diagonales représentent l'énergie de chaque fréquences pour les signaux croisés entre eux. Sachant qu'une valeur singulière est définie pour un pas de fréquence $(k\Delta f)$ alors les fréquences pour lesquelles les valeurs singulières sont maximales représentent les fréquences propres du système. $\underline{\underline{G}}$ est une matrice carré on peut donc se demander pourquoi ne réalisons nous pas simplement une décomposition en valeurs propres ? En pratique la décomposition en valeurs n'est pas réalisée car elle est difficile numériquement à cause des contrastes entre des valeurs singulière très petites correspondant au bruit et très grandes correspondant aux modes.

La troisième étape consiste donc à récupérer les fréquences pour lesquelles les valeurs singulières sont maximales. L'avantage de cette méthode est qu'elle est capable de visualiser si il y a des modes doubles. C'est à dire qu'à une fréquence donnée il existe plusieurs déformées modales. En pratique la première valeur singulière est bien plus grande que les autres excepté en cas de modes doubles. Donc après s'être assuré qu'il n'existait pas de tels modes l'étude de la première valeur singulière suffit à estimer les fréquences propres de la structure. Pour obtenir la déformée il faut s'intéresser à la matrice $\underline{\underline{U_k}}$ pour les k pour lesquels les valeurs singulières sont maximales. Car cette matrice représente les "directions de plus grandes variations". Dans notre cas si un capteur X_i est disposé à une abscisse y_i la donnée de $U_{i,1}$ permettra d'obtenir la valeur de la déformée à cette abscisse.

Ehanced Frequency Domain Decomposition (EFDD) ou décomposition en domaine fréquentiel renforcée

La FDD permet de déterminer les fréquences et les déformées propres de chaque fréquence. L'EFDD permet en plus de déterminer les coefficients d'amortissements modaux. Il s'agit d'une amélioration de la FDD. Après avoir repérer les pics des valeurs singulières (leur valeur en abscisse étant une fréquence propre), il faut sélectionner une largeur du pic. Donc sélectionner un nombre de termes K, de la valeur singulière considérée, proches du pics (K représente la largeur du pic). Puis il faut effectuer une transformée de Fourier inverse de ces valeurs singulières. La transformée de Fourier inverse permet de récupérer un signal de réponse fréquentielle impulsionnelle. En appliquant la méthode du décrément logarithmique à ce signal on obtient l'amortissement pour cette fréquence propre. Ainsi grâce à l'EFDD on obtient la description complète du mode.

Chapitre 4

Application de la frequency domain decomposition (FDD)

Dans ce chapitre nous allons mettre en application la méthode de frequency domain decomposition sur les mesures réalisées sur la dalle de ventilation du tunnel du Siaix en Savoie. Deux campagnes de mesures ont été réalisées. La première campagne a été faite en 2017 et la seconde en 2019 durant mon stage. Des mesures ont été réalisées pendant trente minutes à quatre points du tunnel. Chaque point correspond au creusement d'un rameau entre le tube principal et la galerie de sécurité. Les points sont nommés par la distance par rapport à l'entrée du tunnel, ainsi les quatre points sont nommés PM200, PM380, PM570 et PM741(PM pour point métrique).

En 2017, les mesures ont été réalisées à l'aide de sept capteurs prêtés par le laboratoire de génie civil et de bâtiment (LGCB) de l'ENTPE (je remercie encore le laboratoire qui m'a permis d'avoir accès à ces capteurs). Ces capteurs furent déposés sur la poutre comme présenté en figure 4.1. Ces capteurs sont des vélocimètres de la marque *Tromino*. Chaque capteur acquiert les données dans trois directions notés L, T et V. Ces directions sont notés sur cette même figure (L : Longitudinal aux capteurs, V : Vertical, T : Transversal). Les capteurs ont donc été déposés

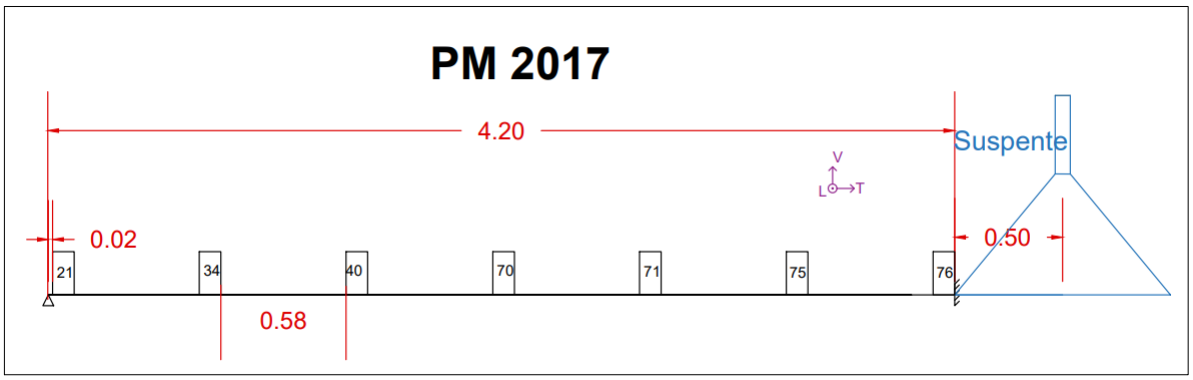

Figure 4.1 – Plan capteurs 2017

sur un côté de la dalle de ventilation. On verra par la suite les limites de cette disposition.

En 2019, les mesures ont été réalisées à l'aide de douze capteurs. Ils furent déposés comme sur la figure 4.2. Le nombre de capteurs est plus important car entre temps le CETU (organisme où j'effectue mon stage) a fait l'acquisition de quatre nouveaux capteurs (ceux-ci ont été alliés aux capteurs du LGCB). Le capteur 34 n'a pas pu être exploité par la suite du fait d'un problème de synchronisation avec les autres capteurs.

A noter qu'au point métrique 200 nous avons effectué des mesures sur les deux côtes de la dalle de ventilation. Le plan des capteurs au PM 200 en 2019 est disponible ci-dessous. Cette

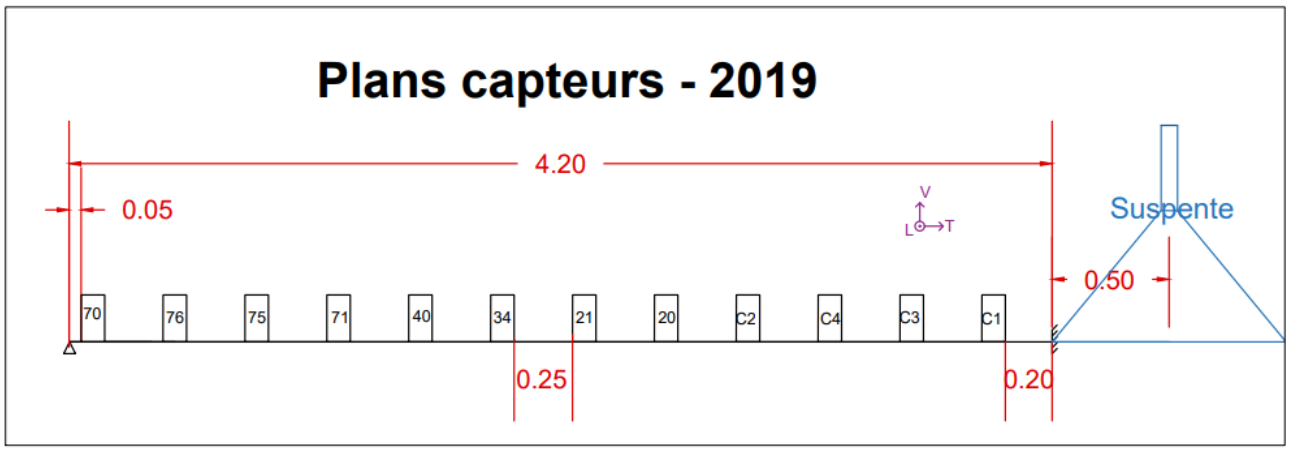

FIGURE 4.2 – Plan capteurs 2019

disposition a été mise en place car certaines déformées modales obtenues à partir des mesures de 2017 n'étaient pas suffisantes pour identifier correctement la déformée analytique obtenues.

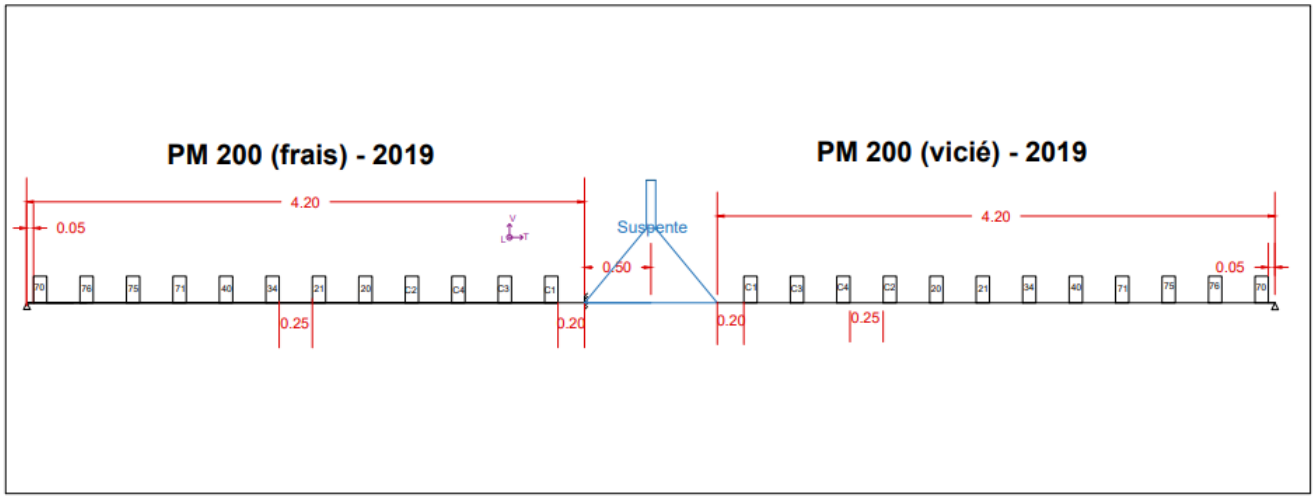

FIGURE 4.3 – Plan capteurs point métrique 200 (2019)

4.1 Peak Picking appliqué à un cas d'étude simple

Avant de se lancer dans la détermination des modes propres de la structure, nous avons cherché à comprendre et à réussir à les mettre en application sur des cas simples. Le Peak Picking n'est pas utilisé pour récupérer les fréquences propres de la structure. Cependant les méthodes qui permettent de déterminer la densité spectrale de puissance en Peak Picking (périodogramme, corrélogramme) sont ré-utilisés pour déterminer les termes de la matrice de densité spectrale croisée dans la méthode de FDD. Ainsi il est intéressant de voir les résultats de cette méthode sur un cas simple d'étude.

Nous avons considéré un signal simple constitué de deux sinus d'amplitudes différentes et d'un bruit gaussien qui brouille les deux sinus. L'idée est de récupérer les fréquences des deux sinus de manière assez précise grâce à la méthode du relevé de pics. La fonction de temps du signal analogique est la suivante, $s : \ t \longmapsto 10\sin(2\pi f_1 t) + 1.5\sin(2\pi f_2 t) + 0,5(bruit\ gaussien)$.

En l'occurrence les fréquences f_1 et f_2 choisies sont 50 Hz et 120 Hertz. Le bruit gaussien est d'espérance nulle et de variance égale à 0,8. D'un point de vu informatique on a crée une fonction *signal*1 qui calcule la valeur de la fonction s pour un t donné. Ensuite on crée deux listes. L'une qui répertorie tout les temps t on l'on évalue la fonction s. L'autre qui répertorie toute les valeurs de s par rapport aux valeurs de t présentent dans la première liste. Ainsi si l'on trace le signal sur 2000 points avec une fréquence d'échantillonnage de 512 Hz on obtient comme signal :

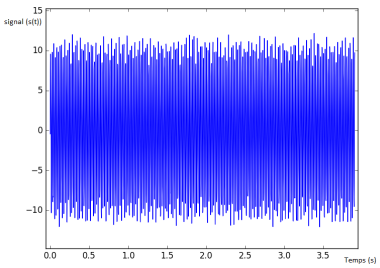

FIGURE 4.4 – Signal test

On utilise trois méthodes différentes pour traiter ce signal, le périodogramme, le corrélogramme et périodogramme lissé (méthode de Welch). Pour tracer ces trois fonctions nous utilisons plusieurs fonctions définies dans la bibliothèque scipy sur python. Même si la programmation de fonctions calculant les valeurs grâce aux formules définies dans le chapitre précédente se est possible, nous utilisons des fonctions de la bibliothèque scipy car celles-ci sont optimisées et permettent un gain de temps non négligeable. Le détail du code pour le calcul des différentes estimées de la densité spectrale de puissance sont accessibles en annexes.

4.1.1 Périodogramme de la fonction test

Pour tracer le périodogramme nous utilisons la fonction *scipy.signal.periodogram*. En entrée de la fonction on renseigne la liste de valeurs du signal évalué à divers instant t et la fréquence d'échantillonnage (içi égale à 512 Hz). Cette fonction renvoie deux listes, une liste de fréquences discrétisées et une liste de valeurs de la densité spectrale de puissance pour ces valeurs. Ensuite on utilise la bibliothèque matplotlib pour réaliser les tracé de courbes (Etant donné que l'on échantillonne à 512 hertz on ne s'interressera qu'aux fréquences allant jusqu'à 256 hertz pour respecter le critère de Nynquist-Shannon). On a tracé trois spectres (figure 4.5). Les deux premiers correspondent au spectre du signal décrit précédemment, l'un est en échelle classique l'autre est en échelle logarithmique. La troisième figure correspond au periodogramme du signal $s : \ t \longmapsto 5\sin(2\pi f_1 t)+1.5\sin(2\pi f_2 t)+2(\textit{bruit gaussien})$, c'est à dire que l'amplitude du premier sinus a été diminué et l'amplitude du bruit gaussien a été amplifié.

On remarque que l'on visualise assez bien les deux fréquences des sinus sur la figure en échelle classique. Cependant en échelle logarithmique on observe des variations, ces variations peuvent poser problème lors de la recherche des pics par la suite. En effet les variations locales, dues au modèle, peuvent être prises pour des pics (ces variations sont à l'origine du biais de l'estimateur). Il est tout de même assez rapide de relever les pics des deux fréquences. Sur la figure du signal plus fortement bruité on distingue toujours bien les fréquences même si l'écart avec la valeur des pics parasites diminue.

4.1.2 Corrélogramme de la fonction test

Pour tracer le corrélogramme, nous utilisons la fonction *scipy.signal.correlate* qui à partir de deux séries de données nous renvoi une liste des valeurs de la fonction de d'inter-corrélation (ainsi si l'on mets les deux mêmes séries de données cette fonction calcule la valeur de la fonction

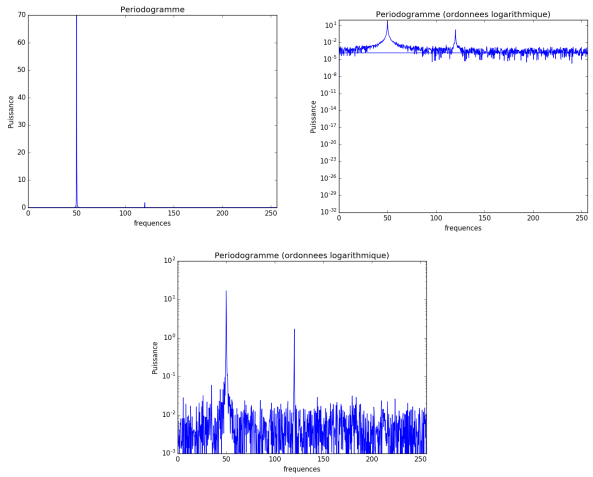

FIGURE 4.5 – Périodgrammes du signal test

d'auto-corrélation). Puis nous effectuons la transformée de Fourier discrète de la liste obtenue grâce à la fonction *scipy.fftpack.fft*. Par définition les valeurs de cette transformée de Fourier correspond aux valeurs du corrélogramme. Puis nous effectuons le tracé (figure 4.6).

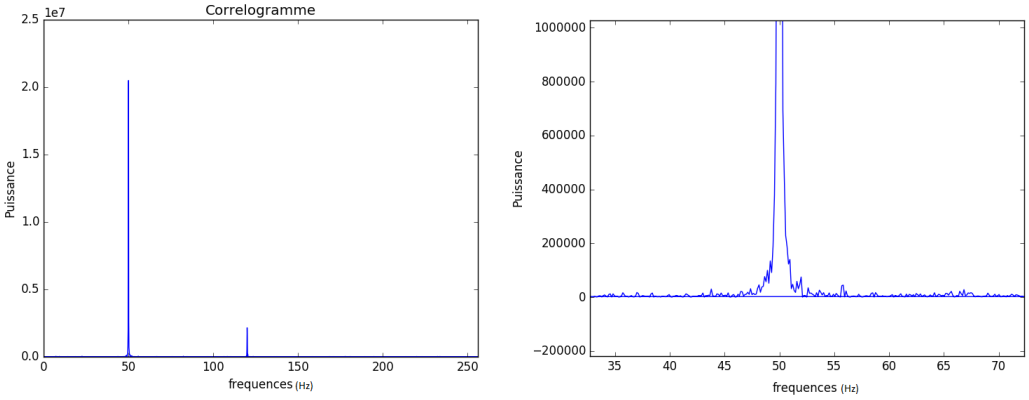

FIGURE 4.6 – Corrélogramme de la fonction test

Les valeurs de la densité spectrale de puissance estimées par corrélogramme sont bien plus élevée que celles du périodogramme (mais leur importance n'est que relative). Du fait des valeurs importantes que prend la densité spectrale de puissance le tracé en échelle logarithmique apporterait peu d'informations. A droite du corrélogramme, il s'agit d'un zoom au niveau du premier pics. On remarque donc que localement il existe des variations comme pour le périodogramme.

4.1.3 Périodogramme lissé ou de Welch

Pour tracer le périodogramme lissé nous utilisons la fonction *scipy.signal.welch*. Pour utiliser cette fonction il est nécessaire de renseigner la liste des valeurs, la fréquence d'échantillonnage, le fenêtrage utilisé et la longueur de ce fenêtrage. Le fenêtrage utilisé est le fenêtrage de Hann. C'est à dire qu'on va convoluer le signal avec la fonction fenêtre de Hann suivante,

$$w(t) = \frac{1}{2} - \frac{1}{2}\cos(2\pi\frac{t}{T}) \ \forall t \ \in [0, T], 0 \text{ sinon}$$

La longueur du fenêtrage sera de 200 points car la longueur de notre signal est de 2000 points et ainsi le critère de longueur du fenêtrage qui doit être 10 fois inférieur à la longueur du signal est bien respecté.

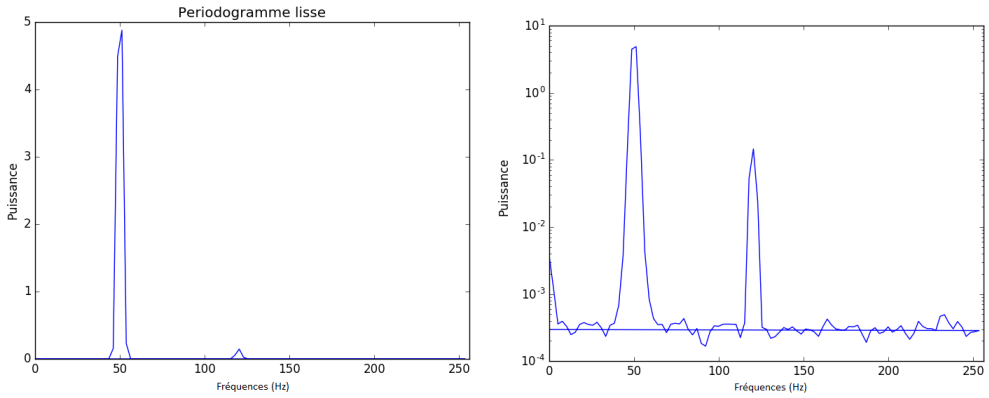

FIGURE 4.7 – Périodogramme lissé du signal test

On remarque que sur ce périodogramme (figure 4.7) les variations sont moins importantes. Cela vient du fait que la variance du périodogramme de Welch est bien moins importante que celle du périodogramme classique. Ce genre de traitement est bien plus exploitable par la suite. Les pics aux fréquences sont un peu plus larges mais les fréquences propres restent bien visibles.

Periodogramme lissé appliqué à une voie d'un capteur

Nous avons observé les résultats des différentes méthodes d'estimations de la densité spectrale de puissance sur un signal que nous connaissions. Ainsi même si les fréquences que nous cherchions étaient plus "remarquables" que les autres nous savions ce que nous cherchions. Dans le cas de l'analyse modale opérationnelle nous ne pouvons avoir qu'une idée des modes que nous allons trouver grâce à une modélisation analytique ou numérique (sujet de la partie 1). Chaque modélisation a ses limites et ses résultats ne sont que des approximations de la réalité. Réalité qui ne peut elle même qu'être approchée par le traitement des données. Ainsi dans ce cas de manière arbitraire nous essayons le périodogramme de Welch sur le signal vertical du capteur 21 de la campagne de 2017 au point métrique 741.

Pour traiter les données relevés nous utilisons la bibliothèque 'pandas' qui s'occupe du traitement des bases données. Chaque enregistrement dure plus de 25 minutes avec une fréquence d'échantillonnage de 512 hertz. Ainsi tous les signaux récupérés (trois par capteurs) sont composés d'au minimum 768 000 points. Avec un tel nombre de données, il est primordial de prêter attention au temps de calcul. C'est pourquoi nous utilisons des fonctions optimisées de la bibliothèque 'scipy'. Les temps de calcul pour une voie par méthode Welch restent négligeable. Ce qui peut prendre du temps est la création de la base de données à travers la bibliothèque 'pandas' (cela reste de l'ordre d'une dizaine de secondes maximum par capteur). Les calculs sont réalisés à l'aide de deux processeurs Intel(R) Xeon (R) CPU de fréquences 2.67 GHz.

Sur le periodogramme lissé en échelle d'ordonnées classique on observe bien les pics jusqu'à 70, 80 hertz, seulement après 100 hertz il est difficile de distinguer les fréquences propres, d'où l'importance de l'utilisation de l'échelle logarithmique. Sur l'échelle logarithmique, plusieurs pics sont observables après 100 hertz. En considérant que chaque pic correspond à une fréquence propre on obtient 20 fréquences propres dont 16 comprises entre 0 et 200 hertz. Or lors de la détermination des modes analytiquement nous obtenions environ une dizaine de pics entre 0 et 200 Hertz. Ainsi certains pics peuvent ne pas correspondre à de réelles fréquences propres comme pour le cas test ou bien notre modèle analytique n'a pas pu prévoir toutes les fréquences propres

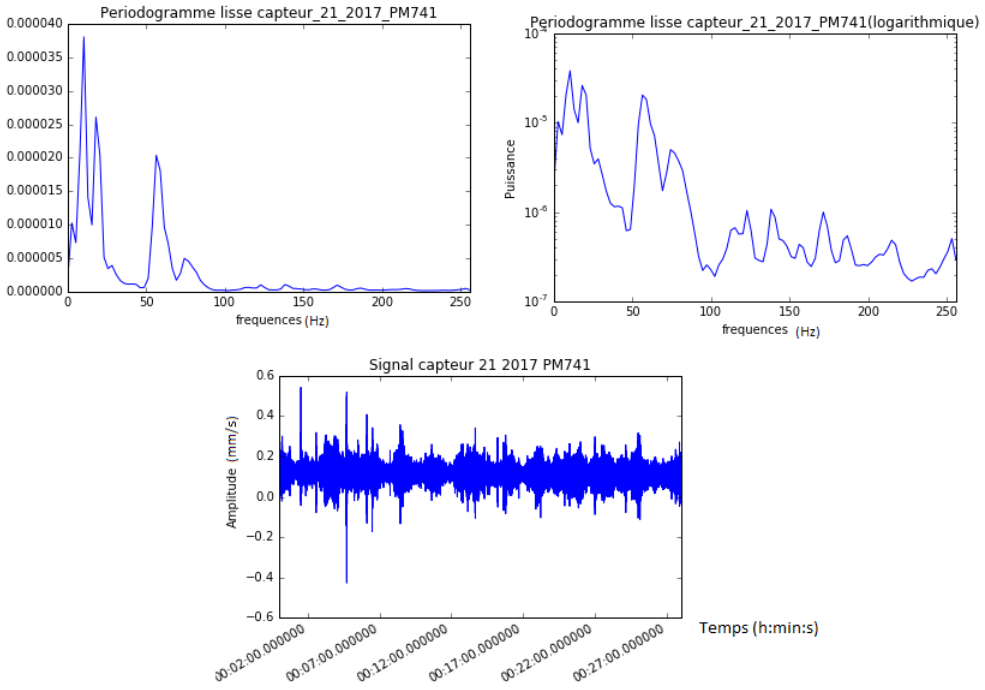

FIGURE 4.8 – Périodogramme de Welch du la voie verticale du capteur 21 de la poutre 741 (2017), en bas le signal récupéré en fonction du temps

possibles. Enfin il se peut que ce capteur soit situé à un noeud de la déformée d'une fréquence propre et qu'ainsi une fréquence propre ne soit pas répertoriée. Enfin la donnée d'un seul capteur ne suffit pas à tracer la déformée. On peut cependant récupérer des valeurs d'amortissements. En effet, en effectuant une transformée de Fourier inverse des valeurs à proximité des pics nous obtenons une réponse impulsionnelle et nous pouvons ainsi déterminer l'amortissement par méthode du décrément logarithmique sur ce signal.

Après avoir mis en pratique les différentes méthodes de bases de traitement de signaux aléatoires stationnaires nous observons que ceux-ci atteignent rapidement leur limites en cas de signaux bruités et de signaux plus complexes. De plus ces techniques permettent d'analyser un signal à la fois. Or lors de nos mesures nos avons 21 signaux (7 capteurs et 3 signaux par capteur) par poutres à traiter pour la campagne de 2017 et 33 signaux par poutre pour la campagne de 2019. Traiter tous les signaux un à un s'avérerait très long et fastidieux pour obtenir des résultats peut significatifs. C'est pourquoi nous utilisons la frequency domain decomposition.

4.2 Frequency Domain Decomposition appliquée au cas d'étude

L'objectif de cette partie est de déterminer avec la méthode de la frequency domain decomposition les modes propres des poutres étudiées. Nous allons nous intéresser à une poutre en particulier pour expliquer la démarche. Étant donné que nous avons réalisé des mesures plus poussées au point métrique 200 en 2019 cette poutre sera choisie. Tous les programmes évoqués pour déterminer les modes propres de la structures seront disponibles en annexes. Nous allons donc nous intéresser à la poutre 'PM200' pour les campagnes de mesures 2017 et 2019. La détermination des fréquences propres et des déformées se fait en trois étapes : La détermination de la matrice de densité spectrale croisée, la décomposition en valeurs singulières et la détermination des fréquences propres à partir de la première valeurs singulière. Dans un second temps nous déterminerons les amortissements associées à chaque fréquences par EFDD.

Détermination de la matrice de densité spectrale croisée

Pour cela nous allons tous d'abord récupérer les différents signaux acquis et les classer dans une matrice regroupant tous les signaux acquis. Pour la campagne de 2017 cette matrice compte donc 21 lignes de N colonnes (N nombres de points d'acquisition). Ensuite une fois cette matrice réalisée nous pouvons calculer la matrice de densité spectrale croisée grâce à la fonction *scipy.signal.csd*. Cette fonction calcule la densité spectrale croisée de deux voies, la méthode de calcul est la même que pour le périodogramme de Welch. Nous devons spécifier la longueur des segments de fenêtrage. Dans notre cas nous divisons la longueur totale du signal par un nombre de portions. Dans un premier temps nous divisons le signal en 256 portions. La matrice de densité spectrale croisée a la même forme que celle en chapitre un de cette partie. Il s'agit donc d'une matrice $21 \times 21 \times N_{2017}$ pour la campagne de 2017 et de taille $33 \times 33 \times N_{2019}$ pour la campagne de 2019. Le détail du code pour calculer les matrices de densités spectrales croisées sont disponibles en annexes. Le fenêtrage utilisé est celui de 'Hann'.

La décomposition en valeurs singulières

A partir de la matrice de densité spectrale croisée d'une poutre nous déterminons la décomposition en valeurs singulières de cette même matrice. Ce calcul est effectué à l'aide de la fonction *np.signal.svd*. Il faut réaliser une décomposition en valeurs singulières de chaque matrice plan k(qui correspond à la valeur de la matrice de densité spectrale de puissance à un pas de fréquence $k\Delta f$) de la matrice totale. Cette fonction permet de récupérer trois matrice **U, V, S**. **S** est la matrice qui répertorie les valeurs singulières sur sa diagonale. Ainsi $S(1, 1, k)$ est la valeur de la première valeurs singulières à la fréquence $k\Delta f$. Pour pouvoir récupérer les déformées nous récupérons les valeurs de la première colonne de **U**. Nous avons trois valeurs de la déformée par capteur. En effet le premier terme correspond à la déformée selon la direction L, le deuxième selon la direction T et le troisième selon la direction V. En prenant la valeur de la déformée selon V nous pouvons tracer les déformées transversales, selon T les déformées longitudinales et selon L les déformées perpendiculaires à la direction de l'étude (hors-plan). Les déformées selon V et T suffisent à retrouver l'aspect des déformées estimées analytiquement. En effet, les déformées selon V et T correspondent aux déformées de la poutre dans le plan.

En traçant les différentes valeurs singulières simultanément on s'assure qu'il n'y a pas de modes doubles. En l'occurrence pour les poutres étudiées nous n'avons pas relevé de modes doubles. Ainsi le tracé de la première valeurs singulières permettra de relever les modes doubles.

Détermination des fréquences propres de la poutre

En traçant la première valeur singulière de la poutre on visualise toutes les fréquences propres que l'on aurait repéré en analysant le périodogramme lissé des voies de chaque capteurs séparément. Ainsi cela permet un gain de temps considérable pour la visualisation des fréquences propres. En effet la visualisation est simplifié mais la sélection des fréquences propres est délicate. Comme on l'a vu précédemment le périodogramme lissé à de fortes variations. Ainsi si l'on effectue seulement une relève des pics, il est possible que certains pics répertorié correspondent en réalité à des 'fréquences fantômes'. Pour discriminer certaines fréquences nous utilisons 'Modal assurance criterion'(MAC) qui permet de comparer les déformées modales de différents modes. Pour cela nous récupérons le plus haut pic puis nous effectuons le calcul du MAC pour les déformées des autres fréquences. Si le MAC est élevé pour deux fréquences différentes cela signifie que ces deux fréquences ont des déformées très proches. Ainsi après avoir déterminé la fréquence de la valeur maximale de la première valeur singulière. On calcule les MAC de cette fréquence avec toutes les autres fréquences. Si le MAC est élevé alors nous pouvons discriminer cette fréquence. La formule du MAC est pour deux fréquences f_1 et f_2,

$$MAC(f_1, f_2) = \frac{(U_{s1,f_1}^T . U_{s1,f_2})^2}{(U_{s1,f_1}^T . U_{s1,f_1}).(U_{s1,f_2}^T . U_{s1,f_2})}$$

Avec

- U_{s1,f_1} la première colonne de U à la fréquence f_1 (respectivement f_2)
- U_{s1,f_1}^T la transposée de la première colonne de U à la fréquence f_1 (respectivement f_2)

La figure 4.9 représente les deux premières valeurs singulière en fonction de la fréquence de la poutre PM200 pour la campagne 2017 (en bleu il s'agit de la première valeur singulière et en vert de la deuxième valeur singulière).

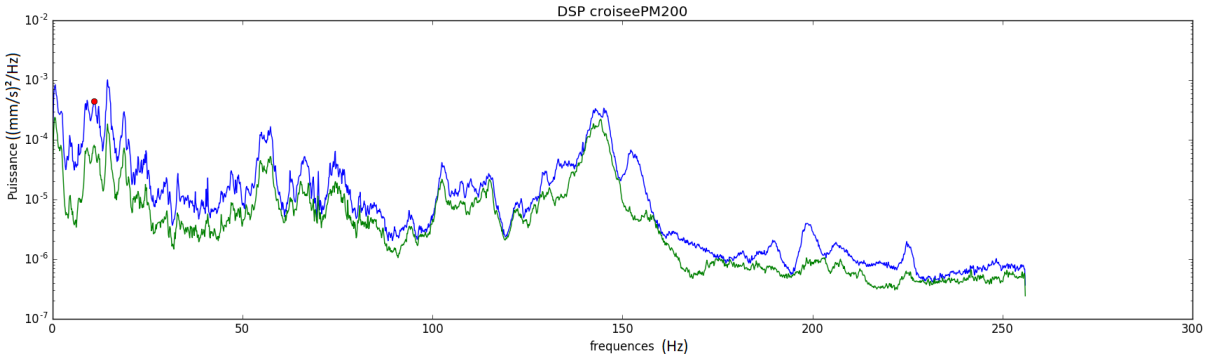

FIGURE 4.9 – Densité spectrale de puissance de la poutre du point métrique 200 de la campagne 2017

On observe une multitude de pics de fréquence. L'idée est d'en récupérer un certain nombre en vérifiant leur 'MAC' et leur déformée pour être sûr de prendre des fréquences réelles (déformées cohérentes) et de ne pas reprendre deux fois le même mode (analyse du 'MAC'). En effet il se peut qu'à certains pics, les déformées associés correspondent à des cinématiques improbables, il convient donc de ne pas les prendre en compte.

On cherche à répertorier trois modes caractéristiques la structure. Ainsi pour la campagne de 2017, le premier mode est à 10,88 Hertz, le second est à 14,54 Hertz (que l'on suppose être les deux premiers modes) et le troisième mode qui nous intéresse est situé à 142,87 Hertz (il s'agit d'un mode en compression). On trace les trois déformées de ces modes, celles-ci sont visibles sur la figure 4.11. Les 'MAC' de ces trois modes sont également visibles à la figure 4.12. On observe que les tracés des 'MAC' des modes 1 et 2 ont des ressemblances. Cela provient du fait qu'ils correspondent au même type de cinématique (Ce ne sont pas les mêmes modes car leurs déformées diffèrent de manière significative). De plus les valeurs du MAC pour les modes 1 et 2 sont élevées dans les fréquences basses et diminue pour des fréquences plus élevées. Ce qui est cohérent car les déformées en hautes fréquences diffèrent fortement des déformées en basses fréquences (même pour le même type de cinématique). Le tracé du 'MAC' du mode en compression montre qu'il n'y a quasiment aucune ressemblance avec les déformées à basse fréquence. Ceci est également cohérent car les modes en compression correspondent à des fréquences plus élevés.

Ces déformées correspondent à des déformées que l'on avait envisagée en partie 1. En effet le premier mode correspond bien à une cinématique en u_m, θ_m. Le deuxième mode peut également être lié à ce type de cinématique. Et le mode à 142,87 Hertz correspond bien au premier mode rigide envisagé. On peut d'ores et déjà noter une différence par rapport aux fréquences estimées analytiquement et numériquement des fréquences relevées opérationnellement. Ces différences peuvent provenir des hypothèses sur les caractéristiques des éléments du modèle (mécaniques et géométriques) et des erreurs liés à la méthode de détermination de ces modes (périodogramme, corrélogramme...).

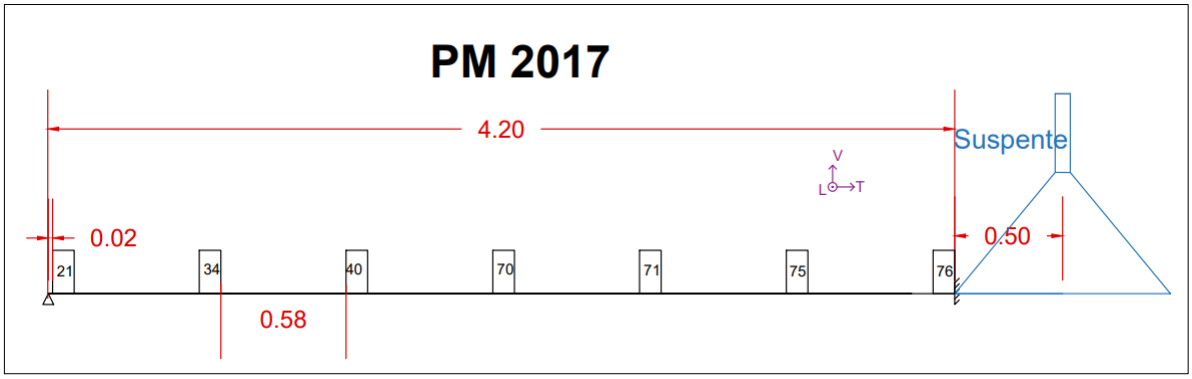

FIGURE 4.10 – Plan capteurs 2017

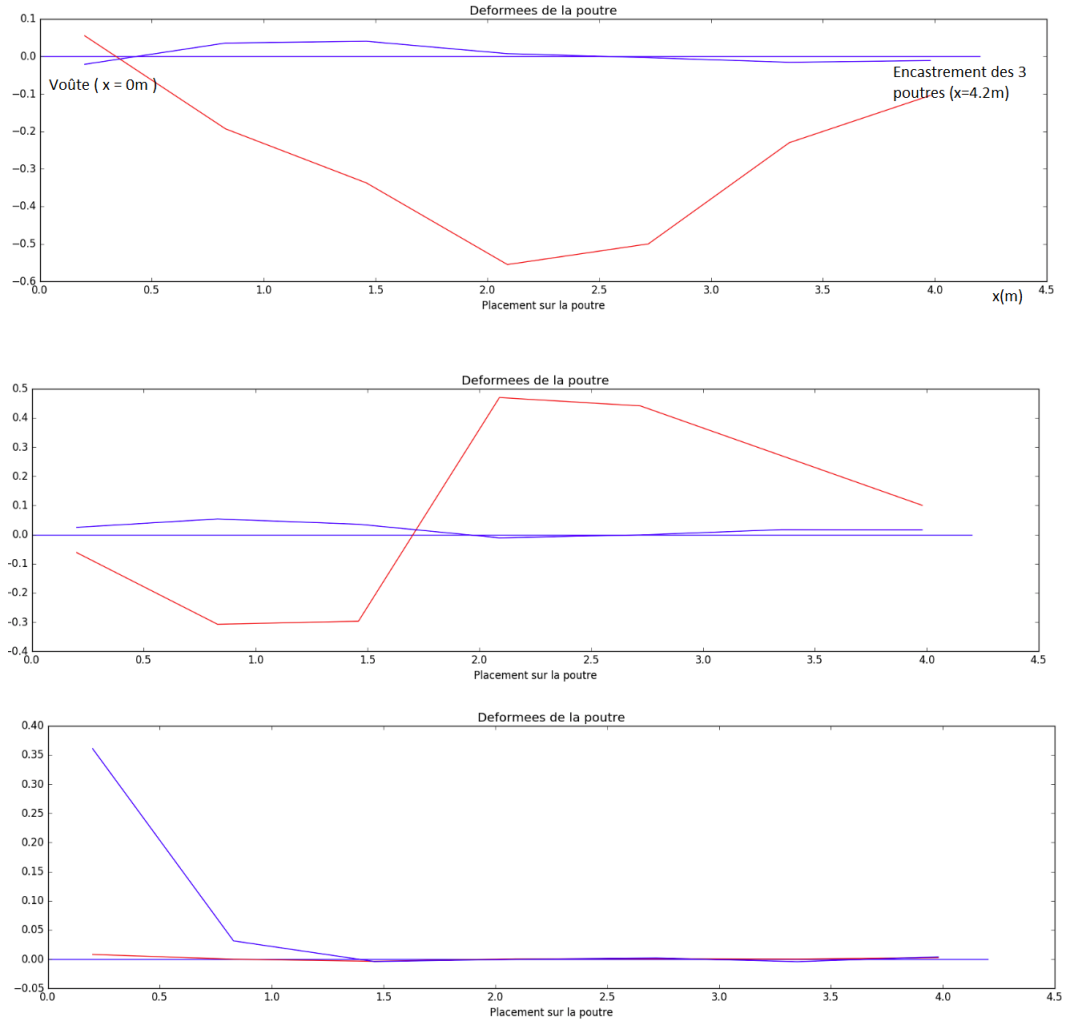

FIGURE 4.11 – Déformations poutre PM 200 en 2017

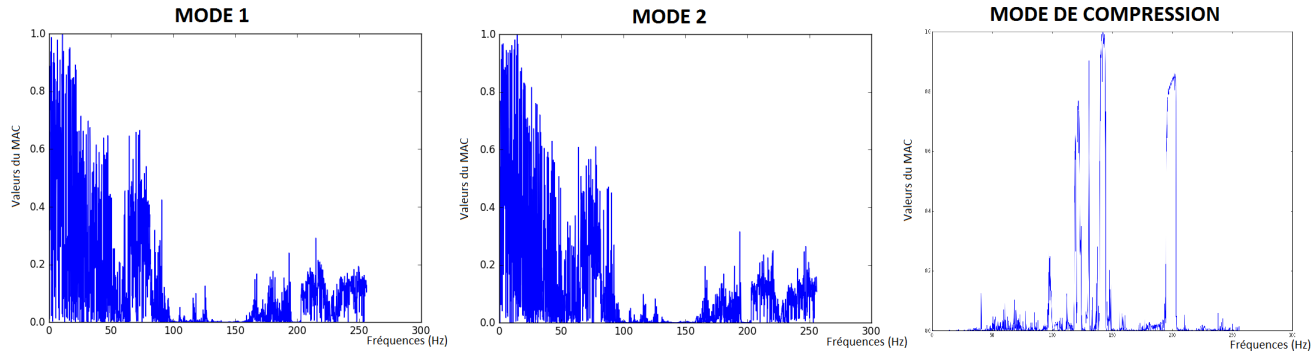

FIGURE 4.12 – Modal Information Criterion des modes PM 200 en 2017

4.3 Ehanced Frequency Domain Decomposition

L'objectif de cette section est de présenter la méthode pour estimer les coefficients d'amortissements propres à chaque fréquence propre. Pour déterminer le coefficient d'amortissement on cible une fréquence propre et on effectue une transformée de Fourier inverse de la première valeur singulière pour des fréquences proches de la fréquence propre considérée. En fait on sélectionne une partie de la première valeur singulière qui correspond à la 'cloche' de la fréquence propre (sur la figure 4.13 on observe la cloche considérée en rouge).

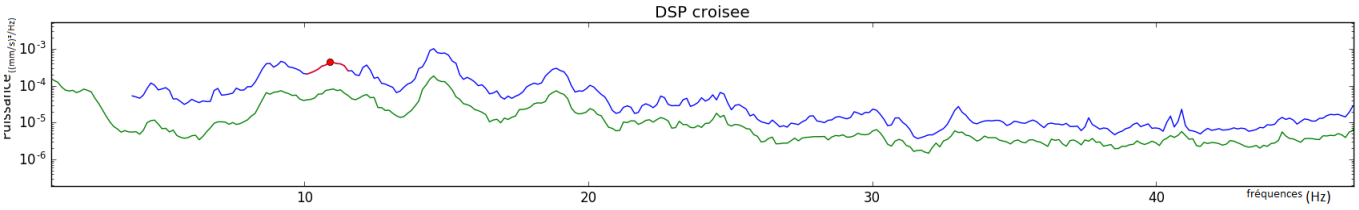

FIGURE 4.13 – Cloche séléctionnée pour faire la transformée de Fourier inverse

Ensuite on réalise la transformée de Fourier inverse de la 'cloche'. Cela nous permet d'avoir une réponse impulsionnelle. En réalisant la méthode du décrément logarithmique sur ce signal on peut retrouver la valeur du coefficient d'amortissement. En effet on a

$$D = \xi_p T$$

avec

- D le décrément logarithmique
- ξ_p le coefficient d'amortissement associé à la fréquence propre f_p
- T la période de la réponse impulsionnelle

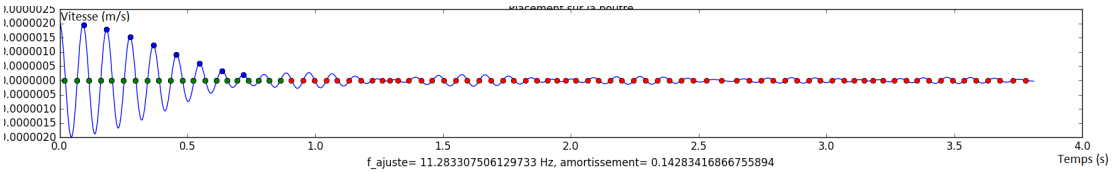

FIGURE 4.14 – Transformée de Fourier inverse de la cloche (calcul de l'amortissement)

Chapitre 5

Application de procédures auto-régressive (AR) et auto-régressive à moyenne mobile(ARMA) pour la détection de modes propres

5.1 Exemple d'utilisation de modèle AR et ARMA

5.1.1 Exemple d'utilisation modèle AR

Pour tester cette procédure, on s'intéresse au même signal test que précédemment. On rappelle que $\forall t \in \mathbb{R}^+,\ s(t) = 10\sin(2\pi 50t) + 1.5\sin(2\pi 120t) + 0.5(bruitgaussien)$. Et de la même façon on cherche à retrouver les fréquences propres (50 et 120 Hz). Pour réaliser le calcul des coefficients auto-régressifs(AR) nous utilisons le module 'time series analysis' de la bibliothèque 'statsmodels' sur Python. Encore une fois ce choix est fait par souci des temps de calcul (les programmes sont accessibles en annexes). Nous avions programmé une fonction (disponible en annexe) qui calculaient les coefficients AR à partir de l'algorithme de Levinson, cependant celle-ci prenait bien plus de temps à calculer les a_k que les fonctions optimisées du module 'time series analysis'.

Comme expliqué en partie 1 le choix de l'ordre est la difficulté majeure des procédures AR. Intuitivement on sait qu'un modèle d'ordre 2 permet d'identifier un mode. En effet, un modèle AR(2) aura deux racines conjugués qui permettent l'identification d'un mode à partir des formules détaillés au chapitre 1 de cette partie. Ainsi puisque dans notre cas d'étude on a deux fréquences propres un modèle AR(4) doit suffir à identifier les deux fréquences propres. Après avoir lancé le calcul nous obtenons les fréquences et amortissements suivants(les paramètres sont calculés grâce à une méthode des moindres carrés) :

Mode	Fréquence (Hz)	Amortissement
1	50.34	0.015
2	135.75	0.1

La première fréquence est proche de 50 hertz cependant son coefficient d'amortissement est non nulle en réalité il devrait être nul, on remarque cependant qu'à mesure que l'ordre augmente l'amortissement diminue. La fréquence du deuxième mode est assez éloignée de 120 Hertz et son coefficient d'amortissement est largement supérieur à l'amortissement de la première fréquence donc ce mode est bien plus amorti. Dans ce cas l'ordre du modèle a été mal évalué. On peut

également tracer la transformée de Fourier des paramètres, cela permet d'avoir une estimation de la densité spectrale de puissance du signal.

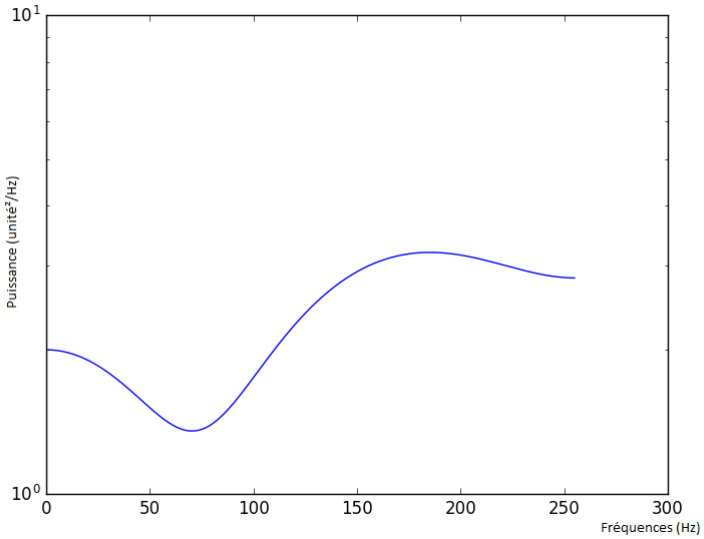

FIGURE 5.1 – Densité spectrale du signal d'une procédure auto-regressive d'ordre 4

Ainsi pour déterminer les fréquences propres à partir d'une procédure AR nous devons au préalable déterminer l'ordre optimal du modèle. Pour cela on va se servir de trois indicateurs : la fonction d'auto-corrélation partielle, le critère d'information d'Akaike et le critère d'information de Bayes. Enfin se servira d'un diagramme de stabilité pour déterminer quelles sont les fréquences propres et amortissements réels.

La fonction d'auto-corrélation partielle d'ordre h permet de mesurer la liaison entre la valeur du signal au rang k par rapport à la valeur du signal au rang k-h une fois retiré les liens entre la valeur au rang k et les valeurs comprises entre k-1 et k-h+1. Ainsi pour un ordre h supérieur à l'ordre réel du modèle la fonction d'auto-corrélation partielle est censée être nulle. En réalité une fois l'ordre du modèle dépassé, la fonction d'auto-corrélation partielle est censée suivre une distribution aléatoire. On étudie donc cette fonction pour savoir à partir de quelle ordre les valeurs oscillent autour de zéro.

En pratique on observe que cette fonction est difficilement exploitable.

Le critère d'information d'Akaike est très utilisé pour déterminer l'ordre d'un modèle cependant celui-ci n'est pas sûr (Olivier Besson, 'Modélisation paramétrique du signal'). Cependant en panachant ce critère avec d'autres indicateurs on peut avoir une estimation de l'ordre à fixer (Même remarque pour le critère d'information de Bayes).

Cependant le calcul de ces indicateurs est très coûteux en temps de calcul même pour des signaux avec un nombre de points faible. C'est pourquoi il faut essayer de trouver un compromis entre temps de calcul et ordre choisi. La détermination des paramètres n'est pas très coûteuse. Les indicateurs d'ordres le sont car ils comparent la prédiction des x_k aux vraies valeurs du signal. Ainsi plus le signal est long plus le nombre de comparaison est élevé et plus l'ordre est élevée plus le calcul des prédictions de x_k sera long (et donc le temps de calcul). Ces indicateurs sont par ailleurs assez peu fiables. En pratique, on utilise plusieurs ordres et on compare les résultats. Le diagramme de stabilité aide à trancher sur les modes à considérer ou non. L'ordre de grandeur de calcul des coefficient d'un modèle AR(400) est de l'ordre de deux heures. L'ordre de grandeur de calcul des indicateurs d'ordre d'un modèle AR(100) est de l'ordre de deux jours et demi.

Le diagramme de stabilité consiste à regrouper sur un plan les modes estimés à différents

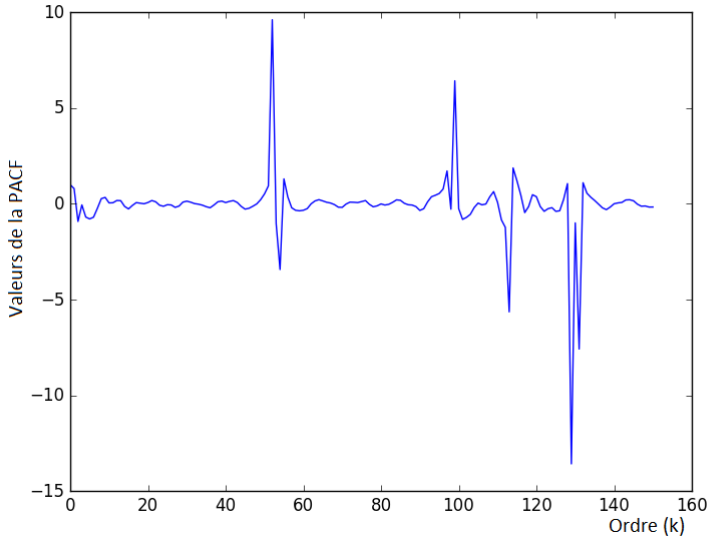

FIGURE 5.2 – Fonction d'auto-corrélation partielle jusqu'à l'ordre 150 du signal test

ordres. En abscisse on a la fréquence du mode et en ordonnée son amortissement. On écarte les modes qui possèdent un coefficient d'amortissement supérieur à 1. Dans notre cas si on trace un diagramme de stabilité jusqu'à l'ordre 40 on obtient le graphique suivant :

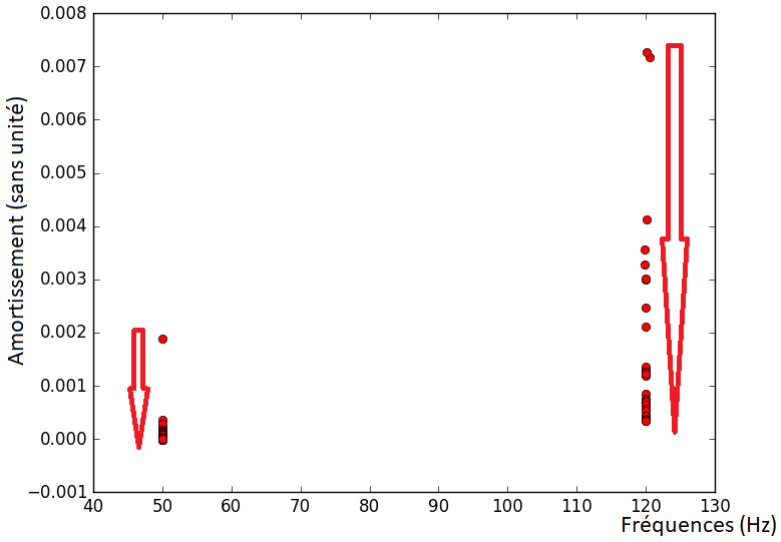

FIGURE 5.3 – Diagramme de stabilité signal test (ordre 40)

Les modes stables sont ceux pour lesquels un nuage de points est regroupé autour de ce mode. Cela signifie que pour des ordres différents ces modes sont à peu près égaux. En l'occurrence dans notre cas on observe directement que deux modes sont présents. Si on prend la valeur de ces modes à l'ordre 40 on obtient les modes ci-dessous.

Mode	Fréquence (Hz)	Amortissement
1	50.0001	$2,6.10^{-5}$
2	120.003	0.02

Ces modes estimés sont bien proches des modes réels. L'amortissement est proche de zéro et les fréquences sont quasiment égales. En traçant l'estimée de la densité spectrale de puissance des paramètres AR à un ordre 40 on obtient la figure 5.4.

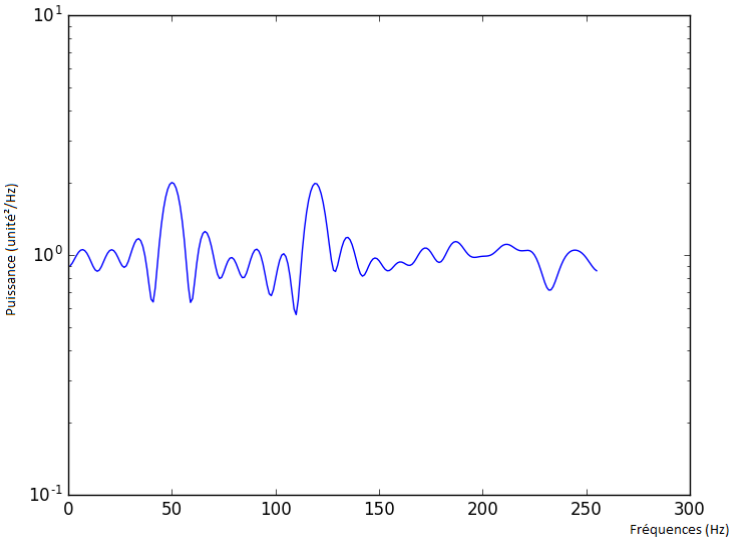

FIGURE 5.4 – Densité spectrale de puissance de la procédure AR d'orde 40 du signal test

Pour ce cas d'école on remarque que le calcul des critères est très couteux en temps de calcul, cependant l'estimation des paramètres AR est relativement rapide. Ainsi il est plus intéressant d'étudier un diagramme de stabilité pour plusieurs ordres et de relever les modes stables plutôt que de calculer des critères peu fiables en pratique.

5.1.2 Exemple d'utilisation d'un modèle ARMA

On garde le même signal test que précédemment. L'avantage des modèles ARMA c'est que le tracé de la densité spectrale de puissance est très lisible. Sur la figure 5.5 on observe le tracé d'un ARMA(6,4) (d'ordre 6 en auto régressif et d'ordre 4 en moyenne mobile) en vert et en bleu le tracé d'un périodogramme. On remarque immédiatement que le tracé du modèle ARMA est bien plus lisible avec un pic à 50 Hertz et à 120 Hertz.

De la même façon que précédemment on peut retrouver les modes du signal, ainsi pour un ARMA(6,4) on a

Mode	Fréquence (Hz)	Amortissement
1	49.999907	$4,75.10^{-5}$
2	119.99993	0.00466
3	0.0	139,53

Le troisième mode n'est bien sûr pas à prendre en compte car il a une fréquence propre de 0 ce qui n'a aucun sens physique et un amortissement extrêmement élevé. Le plus gros problème de ce type de procédure et qu'elle ne converge pas pour tous les ordres et en pratique on remarque pour un ordre supérieur à (6,4) le modèle ne converge pas. Cet aspect des procédures ARMA est d'ailleurs évoqué dans l'article 'Explaining operational modal analysis with data from an arch bridge' (Filipe Magalhaes, Alvaro Cunha).

Ce type de procédure est très coûteuse en temps de calcul dès que le nombre de points devient important. Ainsi pour les signaux il est nécessaire de réduire le nombre de points si l'on souhaite effectuer une procédure ARMA. D'ailleurs si l'on souhaite récupérer seulement les fréquences

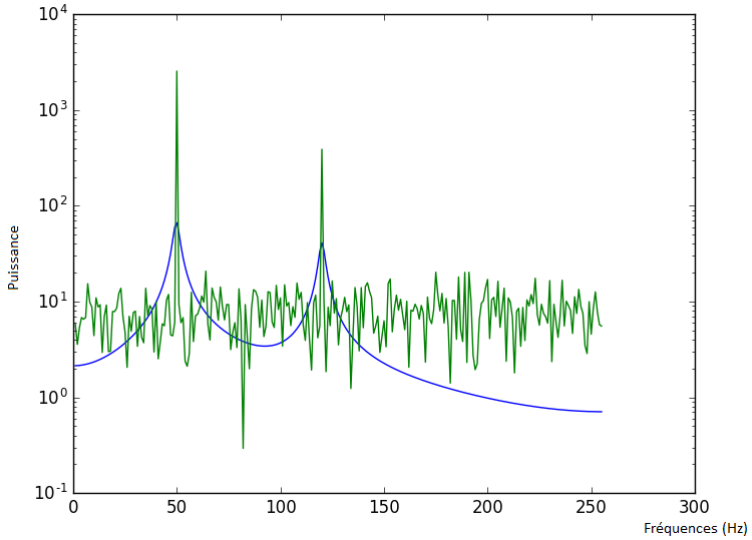

FIGURE 5.5 – Superposition d'une DSP d'un ARMA(6,4) du signal test avec un périodogramme de Welch

propres une procédure auto-régressive suffit. Ainsi pour l'étude on s'intéressera seulement à une procédure auto-régressive.

5.2 Procédure AR appliquée au cas d'étude

5.2.1 Récupération des paramètres AR et identification des modes par diagramme de comparaison

Précédemment on a vu un modèle AR appliqué à un seul signal. Ainsi on peut se poser la question de l'efficacité d'un modèle AR par rapport à la FDD pour le traitement des signaux de plusieurs capteurs. En effet précédemment nous avions expliqué que le peak picking des périodogramme de chaque voie de chaque capteur serait long et fastidieux (c'est pourquoi on lui préférait la FDD). Donc le traitement de chaque voie de chaque signal pourrait être problématique par modèle AR en termes de traitement des données. Cependant contrairement au périodogramme de chaque capteur les modèles AR permettent de récupérer les modes propres (déformées exclues) sans avoir à recourir à une estimation de la densité spectrale de puissance. Ainsi on peut simplement récupérer tous les modes à un ordre donné des capteurs par manipulation informatique. On peut récupérer ensuite les modes propres en observant un diagramme des modes de toutes les voies à un ordre donné sur un diagramme (presque équivalent au diagramme de stabilité).

Si l'on effectue cela pour un ordre 40 pour la poutre 200 en 2017 on obtient la figure 5.6 (en vert, modes verticaux ; en rouge, modes horizontaux ; en bleu, modes tangents) :

Ce graphique est moins exploitable que celui du signal test. Il s'agit du cas réel d'étude donc le nombre de modes est plus important les discerner devient donc plus compliqué. De plus les amortissements sont beaucoup plus élevés que précédemment. Les modes de la structure sont biens amortis (amortissement de l'ordre de 5% pour des structures de génie civil). On peut donc grossièrement écartés certains modes. Cette première méthode atteint rapidement ses limites.

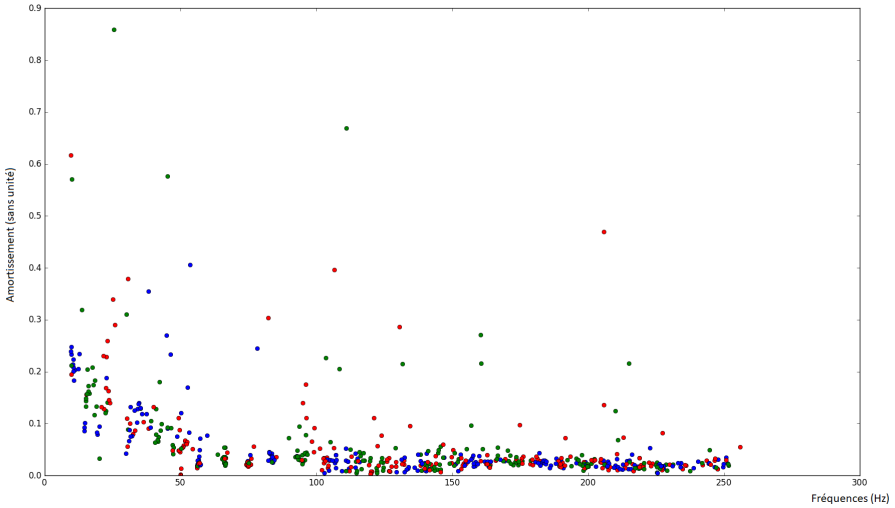

FIGURE 5.6 – Diagramme de comparaison d'ordre des signaux réels récupérés sur un capteur

5.2.2 Utilisation de la FDD sur les paramètres AR pour récupérer les modes de la structure

Précédemment nous avions vu qu'en effectuant une transformée de Fourier des paramètres AR nous obtenions une estimation des la densité spectrale de puissance en fonction de la modélisation auto-régressive. Or la FDD effectue le calcul de la matrice de densité spectrale croisée dans un premier temps puis effectue la décomposition en valeurs singulières de cette matrice. L'idée est de calculer la matrice de densité spectrale croisée à partir des paramètres AR. Cela permettrait d'avoir un spectre plus lisse et donc de repérer plus facilement les modes propres de la structure. De plus en effectuant la FDD nous pourrons également récupérer les déformées des différents modes.

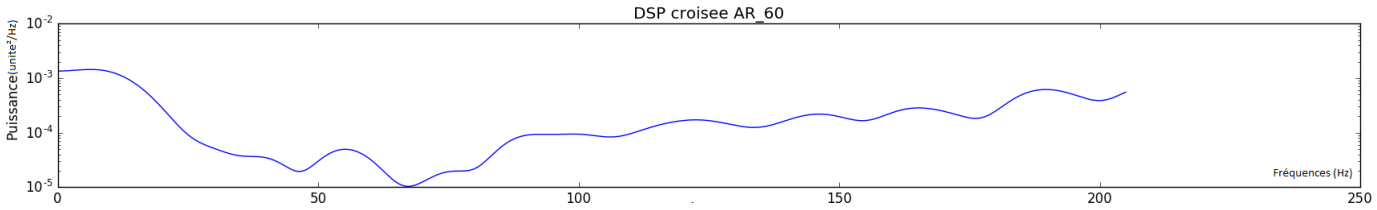

FIGURE 5.7 – Densité spectrale de puissance croisée de tous les paramètres AR d'ordre 60 des signaux récupérés sur la poutre PM200 en 2017

L'avantage de cette DSP est qu'elle permet d'avoir un aspect plus lisse (sans variations locales). L'inconvénient est aussi que cette déformée ne possède pas beaucoup de variations et donc que l'identification de fréquences propres est plus difficile. Cependant en croisant l'analyse du diagramme de comparaison avec cette densité estimée par fdd des paramètres AR on peut récupérer des fréquences propres. Le désavantage est que cette méthode semble bien moins précise que la FDD classique, l'aspect des déformées en témoigne. Pour certains modes des déplacements verticaux non nuls apparaissent près de l'appui simple(fig 5.8). Or ceux-ci devraient être nuls car la poutre est au minimum un appui simple.

Compte-tenu des remarques formulées sur cette méthode, celle-ci semble peu fiable pour estimer les modes propres de la structures (Cela peut être dû à un mode pas assez élevé).

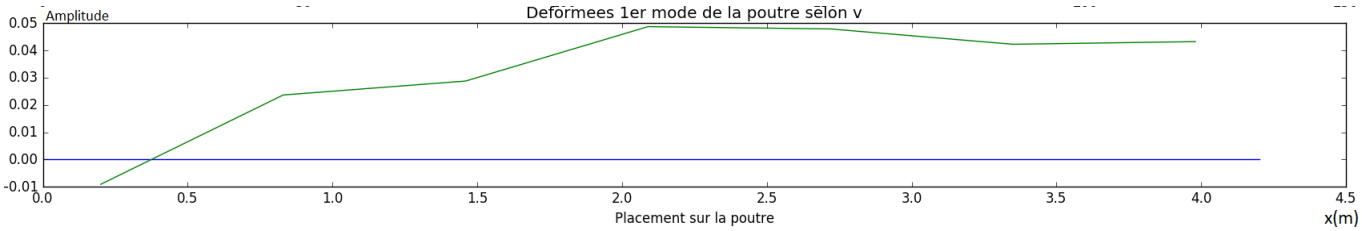

FIGURE 5.8 – Déformée du premier mode de la poutre d'un AR(40) selon la voie verticale

5.2.3 Modélisation auto-régressive appliquée à la transformée de Fourier inverse de la première valeur singulière

Dans le précédent chapitre nous avions vu que l'on pouvait effectuer une transformation de Fourier inverse d'une partie de la première valeur singulière pour obtenir une valeur du coefficient d'amortissement (EFDD). Cependant la FDD ne permettait pas de récupérer avec précision des modes propres de la structure du fait d'un trop grand nombre de variations. L'objectif de cette sous-section est d'arriver à récupérer les modes propres à partir d'une procédure auto-régressive appliquée à la transformée de Fourier inverse du signal.

Après avoir effectué cette modélisation on remarque les résultats sont plutôt mauvais. En effet, on trouve pour certains modes propres des amortissements positifs ce qui n'a absolument aucun sens physique. Nous verrons dans la partie 3 si ces données permettent d'indiquer un endommagement ou non de la structure.

5.2.4 Modèle de Prony moindres carrés

On considère que le signal récupéré peut être décomposé en une somme d'un signal et d'un bruit, soit : $y(n) = x(n) + b(n)$. De plus, le signal est supposé être un signal type exponentielles amorties, c'est à dire :

$$x(n) = \sum_{k=1}^{p} A_k e^{j\phi_k} e^{n(-\alpha_k + j2\pi f_k)}$$

Cette description du signal correspond au modèle de Prony (Olivier BESSON, '*Analyse spectrale paramétrique*'). Cela permet d'assigner à un mode k une amplitude et une phase. Dans le cadre de recherche des modes propres de la structure cela permet de récupérer les amplitudes d'un mode pour chaque capteur et ainsi de possiblement remonter à la déformée modale. On note $z_k = e^{(-\alpha_k 2\pi f_k + j2\pi f_k)\Delta t}$. On peut montrer que les z_k sont les racines du polynôme $A(z) = 1 + \sum_{k=1}^{p} a_k z^{-k}$. Ainsi on a comme pour un modèle AR classique $\alpha_k = \frac{\ln(|z_k|)}{2\pi f_k \Delta t}$ et $f_k = \frac{\arg(z_k)}{2\pi \Delta t}$. Avec les a_k déterminés comme pour un modèle AR classique. Ensuite après avoir effectué le calcul des a_k, on cherche à résoudre le système de Vandermonde surdéterminé suivant :

$$\underline{\underline{Z}}\ \underline{\widehat{h}} = \underline{x}$$

avec

- $Z = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \widehat{z_1} & \widehat{z_2} & \cdots & \widehat{z_p} \\ \vdots & \vdots & \ddots & \vdots \\ \widehat{z_1}^{N-1} & \widehat{z_2}^{N-1} & \cdots & \widehat{z_p}^{N-1} \end{pmatrix}$

- $\underline{\widehat{h}} = \begin{pmatrix} \widehat{h_1} \\ \widehat{h_2} \\ \vdots \\ \widehat{h_p} \end{pmatrix}$ avec $\widehat{h_k} = A_k e^{j\phi_k}$
- $\underline{x} = \begin{pmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{pmatrix}$ les valeurs du signal récupéré.

La résolution de ce système au sens des moindres carrés est $\widehat{h} = (\underline{\underline{Z}}^H \underline{\underline{Z}})^{-1} \underline{\underline{Z}}^H \underline{x}$. $\underline{\underline{Z}}^H$ est la transposée hermitienne de la matrice Z. En appliquant cette méthode au signal test et en renseignant dans un graphique les amplitudes de chaque modes on obtient la figure 5.9.

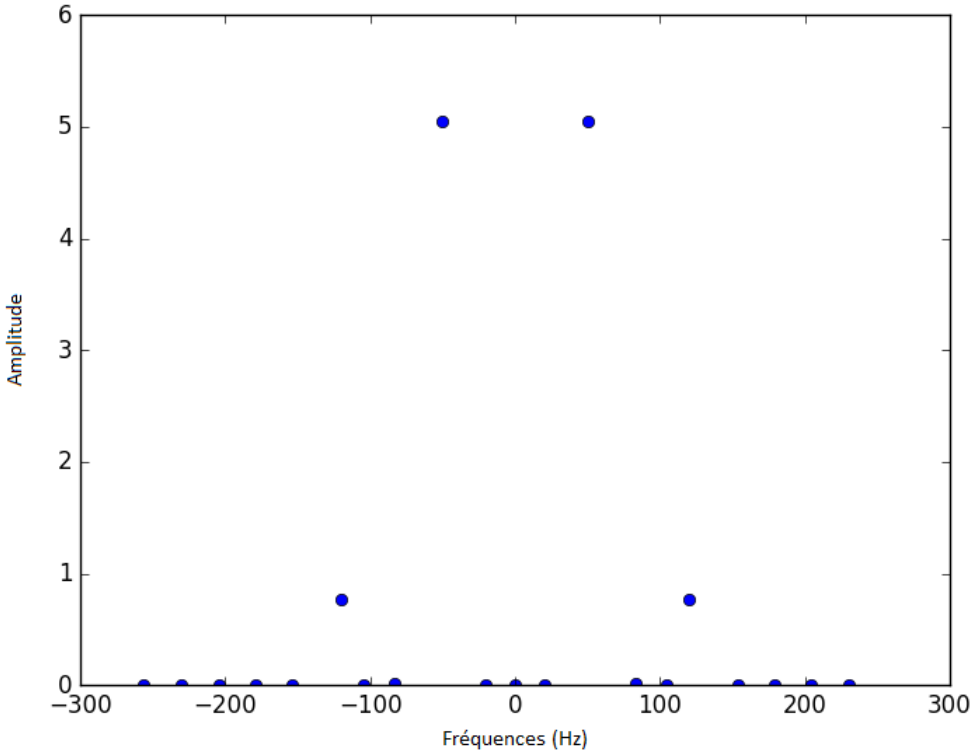

FIGURE 5.9 – Amplitude des fréquences propres du signal test

L'amplitude des modes propres est beaucoup plus importante que les autres modes. Cela permet de récupérer les modes propres. En appliquant ce modèle aux signaux récupérés par les capteurs on obtient la figure 5.10 (pour un ordre AR 60 du signal vertical du capteur 71 de la poutre PM200 en 2017).

On remarque un point d'amplitude élevé à 0. Puis des amplitudes importantes aux fréquences 66.44 Hz, 15.52 Hz, 74.94 Hz, 9.90 Hz. Deux limites sont liés à ce modèle. Premièrement la recherche des amplitudes et des phases (A_k, ϕ_k), dépendent du nombre de points N qu'on va utiliser pour résoudre le système de Vandermonde. Car en théorie pour avoir la meilleure résolution il serait nécessaire de prendre tous les points du signal. En pratique les coefficients de la matrice $\underline{\underline{Z}}^H \underline{\underline{Z}}$ sont trop élevés numériquement pour effectuer le calcul de l'inverse de cette matrice. On a affaire à une limite de représentation des grands nombres. Le second problème est

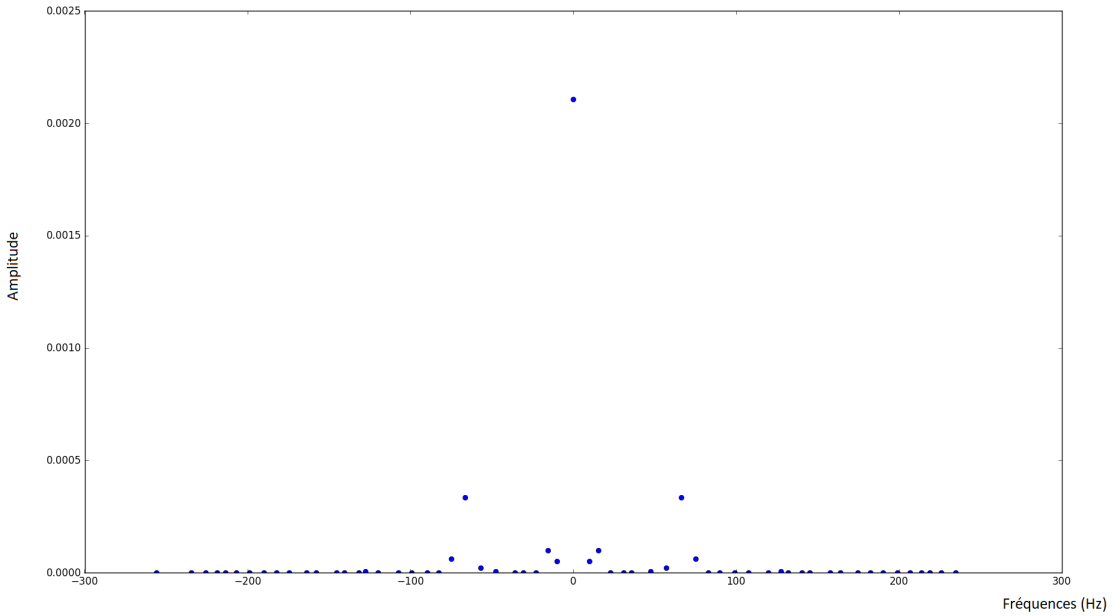

FIGURE 5.10 – Amplitudes des fréquences propres selon la voie verticale du capteur 71 de la campagne 2017 (PM200)

lié à l'ordre choisi. En effet pour différents ordres d'autres fréquences propres apparaissent et certaines déjà existantes évoluent. Cela peut venir du fait que l'on a pas atteint l'ordre optimal. En effet, nous avons effectué plusieurs calculs de l'ordre optimal. Nous l'avons effectué avec divers ordres maximums 20, 40, 60 et 100. A chaque fois l'ordre maximal correspondait à l'ordre optimal. Ainsi il se peut que l'ordre optimal soit bien plus élevé. Or la détermination de cette ordre optimal est trop coûteux en temps de calcul (48 heures pour déterminer l'ordre optimal parmi les 100 premiers ordres).

Nous allons donc utiliser le modèle AR-Prony de façon à récupérer des modes même si nous ne sommes pas sur qu'ils correspondent aux modes exactes de la structure. Cela permettra en complément de la EFDD d'avoir une idée des modes de la structure. De plus nous étudierons les changements entre la modélisation des mesures de la PM200 de 2017 et de la PM200 de 2019.

Troisième partie

Analyse modale opérationnelle appliquée à la dalle de ventilation du tunnel du Siaix

L'objectif de cette partie est de comparer les résultats des différents modèles entre les deux campagnes de mesures. La démarche d'analyse est la suivante. Dans un premier temps, il convient de relever les écarts entre les modèles utilisés pour traiter le signal. Puis de comparer les résultats des campagnes de mesures entre 2017 et 2019. On analysera les données de deux manières différentes. Les données analysées sont celles recueillies au point métrique 200.

La première analyse se fera sur les résultats du traitement d'une voie d'un capteur de 2017 et d'une voie d'un capteur de 2019. Puisque nous ne possédions pas le même nombre de capteurs entre 2017 et 2019, nous n'avions pas la même disposition de capteurs. Ainsi un même capteur ne se trouvait pas à la même place en 2017 et en 2019. Il faut donc essayer de trouver deux capteurs qui se trouvaient quasiment au même endroit sur les 2 campagnes de mesures et qui de plus se trouvaient à un endroit où les modes peuvent être captés facilement(amplitude des modes suffisamment importante). Or le capteur 76 en 2017 était situé quasiment au même endroit que le capteur C1 en 2019. De plus ces deux capteurs sont situés près de l'encastrement des trois poutres (donc soumis à des amplitudes non nulles pour les premiers modes). Ainsi on s'intéressera aux données de la voie verticale du capteur 76 de 2017 et aux données de la voie verticale du capteur C1 de 2019. Cette première analyse permettra de comparer les résultats des modèles utilisés localement tout en évoquant les limites de celle-ci.

La deuxième analyse se fera sur les résultats par méthodes d'analyse multi-capteurs. Les méthodes d'analyses multi-capteurs ont l'avantage de pouvoir donner des déformées de la structure (Les méthodes d'analyse par capteur permettraient de remonter aux déformées cependant cela serait long et fastidieux. De plus l'avantage des méthodes d'analyses multi-capteurs est qu'elles étudie la corrélation des différents signaux entre eux). Ainsi nous pouvons réellement parler d'analyse 'modale' (mode composé du triplet $[f, \xi$,deformées]) dans les méthodes de traitement multi-capteurs. Les deux méthodes employées seront la 'Frequency Domain Decomposition' et des modèles AR-Prony d'ordres élevés. Comme pour l'analyse par capteur on analysera les différences de résultats entre modèles pour une campagne donnée puis dans un deuxième temps les différences entre deux campagnes.

Chapitre 6

Analyse d'un capteur

En partie deux nous avons détaillé plusieurs techniques de traitement de signaux d'une voie. Nous avons dégagé les avantages et inconvénients de chaque méthode de traitement. Nous allons utiliser une méthode d'analyse fréquentielle du signal et une méthode d'analyse paramètrique du signal. La méthode qui sera utilisée pour l'analyse fréquentielle d'un signal sera le 'peak picking'. Or comme on l'a vu lors de la partie précédente, l'estimation des fréquences par 'peak picking' est plus fiable si l'estimation de la densité spectrale de puissance est faite par corrélogramme lissé ou périodogramme lissé. Ainsi nous choisissons comme premier traitement un peak picking sur le périodogramme de Welch de ces signaux. La méthode qui sera utilisée pour l'analyse paramétrique du signal sera un modèle d'AR-Prony. Les modèles auto-régressifs classiques sont écartés car ils ne donnent pas d'informations sur l'amplitude des fréquences. En particulier on étudiera les différences de résultats entre plusieurs ordres (60, 200, 300, 400).

6.1 Campagne de 2017

On rappelle que pour cette campagne on étudie la voie verticale du capteur 76.

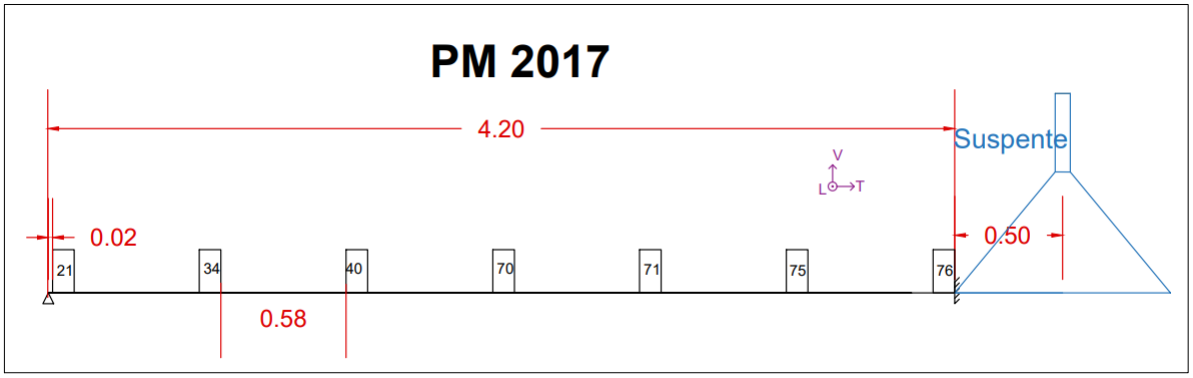

FIGURE 6.1 – Plan capteurs 2017

6.1.1 Périodogramme de Welch

Le périodogramme de Welch est un périodogramme lissé, il possède donc un fenêtrage. Le fenêtrage utilisé est un fenêtrage de Hann. La longueur du signal est de un million de points.

La fenêtre correspond à un nombre de points dans le domaine temporel. Ainsi plus la fenêtre sera longue dans le domaine temporelle plus elle sera étroite dans le domaine fréquentiel et plus l'estimation de la dsp sera précise. Il est donc nécessaire de sélectionner un fenêtrage optimal. Or selon Fabrice HEITZ *'traitement des signaux aléatoires* le fenêtrage optimal doit être le plus

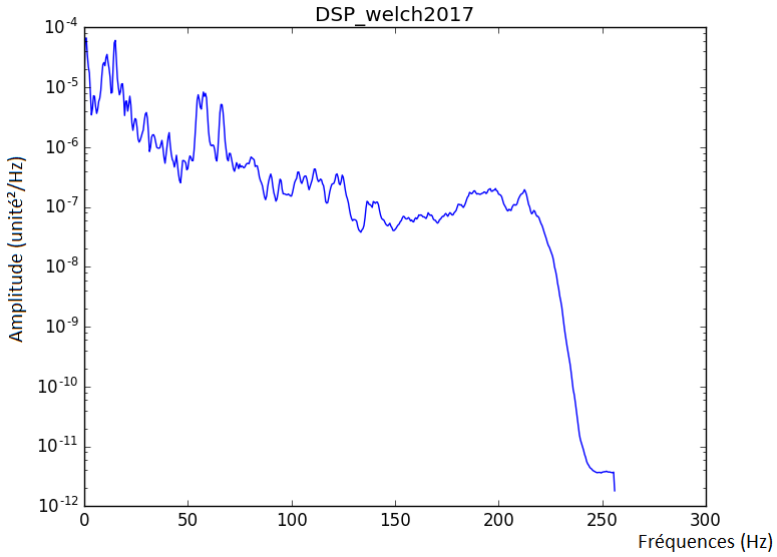

FIGURE 6.2 – Périodogramme de Welch (2017, PM 200, capteur 76, voie verticale). L'amplitude correspond à la puissance de la densité spectrale. Il est renseigné $unite^2.(Hz)^{-1}$ comme unité. Cette unité est de $mm.s^{-1}$ pour la densité spectrale de puissance obtenue par FDD. Cependant pour la DSP obtenue par spectre de Prony tout ce que 'on sait c'est que cette unité est relative à une vitesse. En effet on ne peut qu'avoir une amplitude à un coefficient multiplicatif près ainsi l'ordre de grandeur de l'unité ne peut être identifié clairement. Cependant cela n'a qu'une importance relative car nous cherchons à retrouver les pics de la densité spectrale de puissance.

long possible mais doit rester très inférieur à la longueur du signal (10 fois plus court que la longueur du signal). Dans le cadre de mon travail de fin d'études on choisit un nombre de points en 2^x et on choisit x tel que $2^x \leq 10 \times l_{signal}$ avec $x \in \mathbb{N}$ (la longueur est en 2^x car les fonctions utilisées sur informatique sont optimisées en temps pour de telles longueurs). La densité spectrale de puissance peut paraître brouillée pour des fenêtrage long. Pour que celle-ci paraisse moins aléatoire il faut utiliser un fenêtrage court. Par exemple si on utilise un fenêtrage de 256 points on obtient la densité spectrale de puissance de la figure 6.2. La densité spectrale de puissance est lisible mais le pas est de 0.5 hertz. La sélection des fréquences propres est donc plus facile (lisibilité de la densité spectrale de puissance (dsp)) mais les fréquences séléctionnées seront moins précises. Les fréquences propres obtenus avec ce fenêtrage sont 10.5, 14, 8.5, 18, 5.5 Hertz. Ces fréquences sont estimées par 'peak picking' avec un pas de 2 Hertz. C'est à dire que tous les 2 hertz on relève une fréquence maximale (le détail du programme utilisé est disponible en annexe ...). Les fréquences estimées avec une dsp 'lisible' sont donc moins précise(pas de fréquence large). Il est donc nécessaire de sélectionner un fenêtrage optimal.

Cependant pour une longueur de segments optimale la sélection de pics peut être compliquée car il y a énormément de variations comme on peut le voir sur la figure 6.5. Pour sélectionner les fréquences propres nous utilisons une méthode de 'peak picking'. Pour la campagne de 2017 on a $x = 16$ donc la longueur du fenêtrage est $2^{16} = 65536$ points. Les cinq premières fréquences propres sont 14.79, 0.80, 10.98, 8.70, 18.14 Hertz. On compare maintenant ces fréquences à une modélisation AR-Prony.

6.1.2 Modélisation AR-Prony

L'objectif de cette partie est de sélectionner plusieurs ordres pour la modélisation auto-régressive et de comparer les résultats de ces différents ordres. Ces ordres ont été choisis arbitrairement (en réalité on prend des ordres élevés pour observer si les fréquences convergent).

Ainsi on s'intéresse aux ordres 60, 200, 300 et 400. Ces ordres sont assez élevés car on sait que pour un ordre p le nombre de fréquences propres maximal que l'on trouvera sera de $\frac{p}{2}$. On s'intéressera pour chacun de ces modes aux cinq fréquences propres d'amplitudes maximales et à leur spectre de Prony. Le spectre de Prony ne correspond pas réellement à la densité spectrale de puissance du signal, il correspond à la densité spectrale de puissance de la modélisation du signal. Si on considère que le signal est symétrique autour de l'origine, la valeur de la transformée de Fourier de ce signal est :

$$X(z) = \sum_{k=1}^{p} h_k \left(\frac{1}{1 - z_k z^{-1}} - \frac{1}{1 - (z_k^* z)^{-1}} \right) \text{ avec } z = e^{j2\pi f}$$

. Et par définition la densité spectrale de puissance est $S(f) = |X(f)|^2$. Si on utilise ce spectre pour la densité spectrale de puissance de la fonction test à un ordre 10 on obtient la figure 6.3. On se rapproche de la précision d'un modèle auto-régressif à moyenne mobile sans en avoir les inconvénients (temps de calcul, problèmes de convergence..). De plus nous n'avons pas de fluctuations locales comme dans le cas du périodogramme.

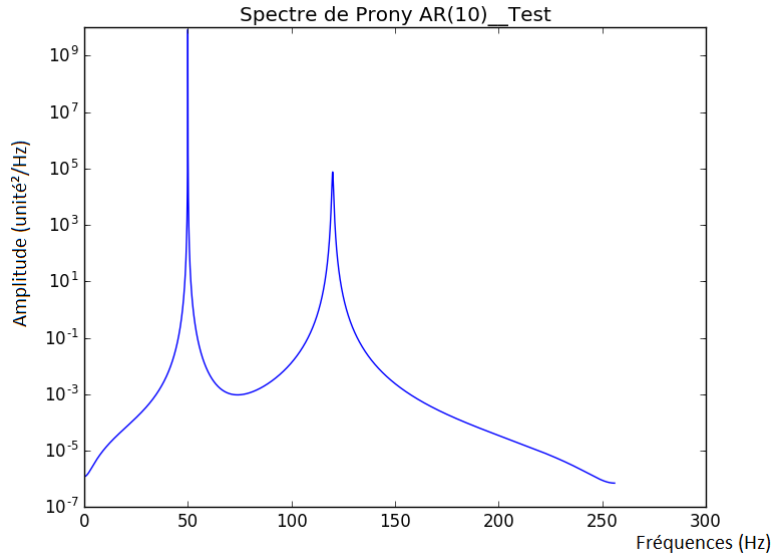

FIGURE 6.3 – Spectre de Prony d'ordre 10 du signal test

On peut donc tracer le spectre de Prony pour les différents ordres et relever leur fréquences propres maximales par 'peak finding'. Les densités spectrales de puissance de ces quatre ordres sont disponibles sur la figure 6.4.

Les cinq premières fréquences propres obtenues sont répertoriés dans le tableau ci-dessous. Celle-ci ont été relevées par recherche de pics avec des pas de fréquences de 2 Hertz (Ces fréquences sont classés par amplitude, chaque fréquence à une amplitude sur la densité spectrale de puissance, les fréquences propres correspondent aux fréquences de plus grande amplitude).

Ordres	Fréquence 1 (Hz)	Fréquence 2	Fréquence 3	Fréquence 4	Fréquence 5
60	9.83	14.99	66.66	57.14	31.11
200	1.23	11.33	14.83	9.0	4.81
300	0.84	10.75	14.72	4.76	7.09
400	0.81	11.0	8.91	14.93	4.7

On remarque que plus les ordres sont élevés plus les fréquences propres paraissent converger (cela est étudié plus en détail dans la partie suivante). Les valeurs de l'ordre 60 sont assez

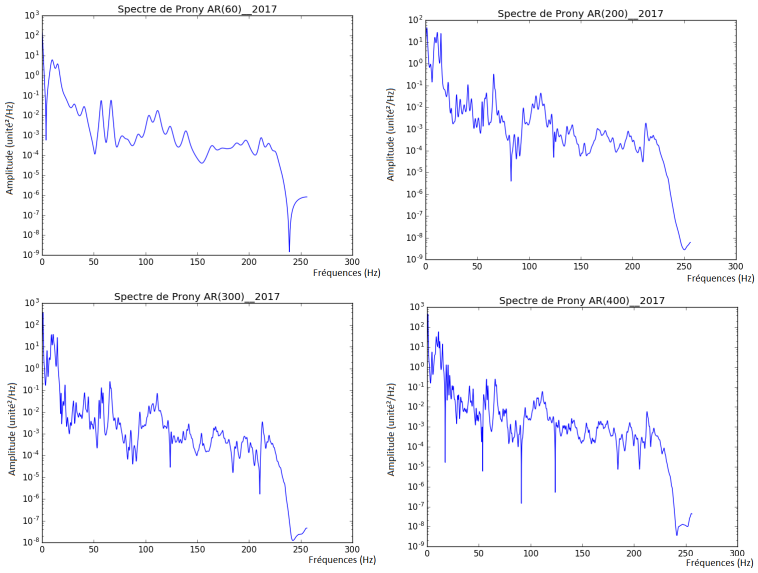

FIGURE 6.4 – Comparaison des différents spectres de Prony pour des ordres différents

différentes des ordres plus élevés. Les fréquences propres des ordres 300 et 400 sont assez proches les unes des autres. Cependant nous avons estimé ces fréquences par 'peak finding' avec un pas de 2 hertz or dans la détermination des modes analytiquement on a vu que certains modes étaient proches. De plus les modèles auto-régressifs nous permettent de récupérer un certains nombre de fréquences propres sans avoir à effectuer une recherche de pics. Cette méthode est utilisée pour comparer les résultats aux valeurs estimées avec le périodogramme de Welch.

6.1.3 Différences entre les méthodes d'estimations des fréquences propres

On va comparer les cinq premières fréquences propres estimées par modèles AR d'ordre 200, 300, 400 et celles estimées par relevé de pics du péridiogramme de Welch. On calculera ensuite l'écart relatif entre ces méthodes sur trois fréquences propres.

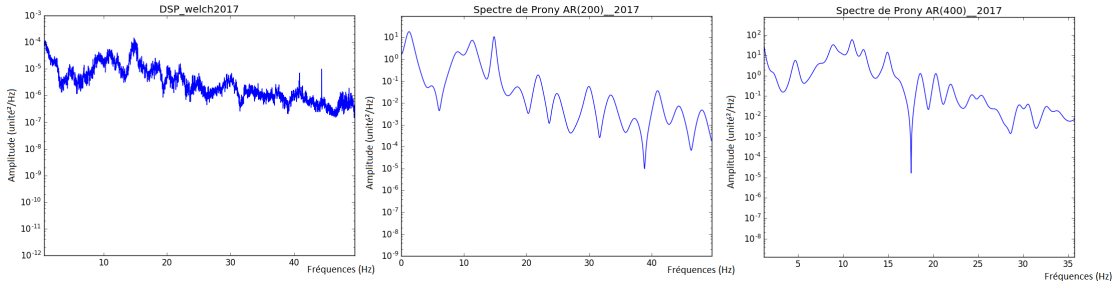

FIGURE 6.5 – (de gauche à droite) Densité spectrale de puissance estimée par méthode de Welch, spectre Prony d'ordre 200, spectre Prony d'ordre 400

Ordres ou Welch	Fréquence 1 (Hz)	Fréquence 2	Fréquence 3	Fréquence 4	Fréquence 5
Welch	14.79	0.80	10.98	8.70	18.14
200	1.23	11.33	14.83	9.0	4.81
300	0.84	10.75	14.72	4.76	7.09
400	0.81	11.0	8.91	14.93	4.7

Les cinq premières fréquences semblent assez proches seulement le classement de ces fréquences n'est pas le même selon les méthodes d'estimations. Quatre fréquences semblent être redondantes. On sélectionne donc les trois premières fréquences propres estimées par relevé de pics du périodogramme de Welch. On les compare aux fréquences propres les plus proches dans les cinq premières fréquences des différents ordre AR.

Ordres	Écart (%) Fréquence 1 = 14.79 (Hz)	Écart Fréquence 2 = 0.80	Écart Fréquence 3 = 10.98
200	$2.7 \times 10^{-}3$	0.35	0.03
300	$4.7 \times 10^{-}3$	0.05	0.02
400	$9.4 \times 10^{-}3$	0.01	$1.81 \times 10^{-}3$

Globalement les écarts (pour l'estimation des 3 premières fréquences) entre les modèles AR et les méthodes d'estimations de Welch semblent diminuer avec l'ordre. L'idée pour déterminer ou non l'endommagement est de comparer les fréquences estimées par modélisation AR et par périodogramme de Welch durant la campagne de 2017 et de 2019. Pour cela on détermine un ordre pour lequel on obtient à peu près les mêmes fréquences que pour un relevé de pics sur le périodogramme de Welch. Un ordre 300 semble convenir. Après avoir déterminé cet ordre on peut éliminer les modes liés à la structure des modes liés au système source-structure à partir d'un critère sur l'amortissement. En effet, les modes liés à une structure de génie civil possèdent un coefficient d'amortissement de l'ordre de 5%. Les trois premiers modes obtenus pour la poutre PM200 de ce capteur sont 8.98, 10.74, 12.01 Hertz. Analytiquement les trois premières fréquences propres étaient de 7.74, 11.44 et 12.35 Hertz. Ces résultats sont relativement proche de ce qui avait été calculer analytiquement. On remarque que la fréquence propre estimée à 0.80 Hertz a été écarté. Son coefficient d'amortissement était de 30%. Cela est rassurant car une fréquence propre aussi faible pour la structure est trop éloignée des résultats des modélisations analytique et numérique.

En effectuant cette même méthode pour ce capteur de la poutre PM380 en 2017 on obtient 14.75, 10.85 et 12.06. La fréquence autour de 9 Hertz est présente dans les 'bons modes' cependant elle possède une amplitude plus faible. Cela peut s'expliquer par la différence de placement exact sur les deux poutres. Pour la poutre PM570 on a 9.71, 11.28 et 12.81. Pour la poutre PM741 on a 9.65, 16.17, 17.07. Entre les poutres les modes principaux ne sont pas les mêmes et ceux-ci pour les deux méthodes d'estimations. Cependant entre les deux méthodes on retrouve quasiment les mêmes résultats. Les caractéristiques des quatre poutres en 2017 ne sont donc pas exactement les mêmes. L'étude précise de ces différences n'a pas été réalisée pendant mon TFE mais ces écarts peuvent s'expliquer par la différences de contexte et de géométries exactes.

6.2 Campagne de 2019

On rappelle que l'on s'intéresse au capteur C1.

6.2.1 Périodogramme de Welch

On garde un fenêtrage de 2^{16} points. La figure 6.7 correspond au périodogramme de Welch obtenu pour le signal de 2019.

Les cinq premières fréquences propres obtenues sont 10.80, 13.90, 8.54, 18.08 et 16.07 Hertz (Celles-sont pointées sur la figure 6.8).

6.2.2 Modèle AR-Prony campagne 2019

On s'intéresse seulement à un modèle Prony d'ordre 300. Les cinq premières fréquences propres sont 14.32, 2.33, 17.69, 5.06, 10.89 Hertz. Trois sont proches des fréquences retrouvées

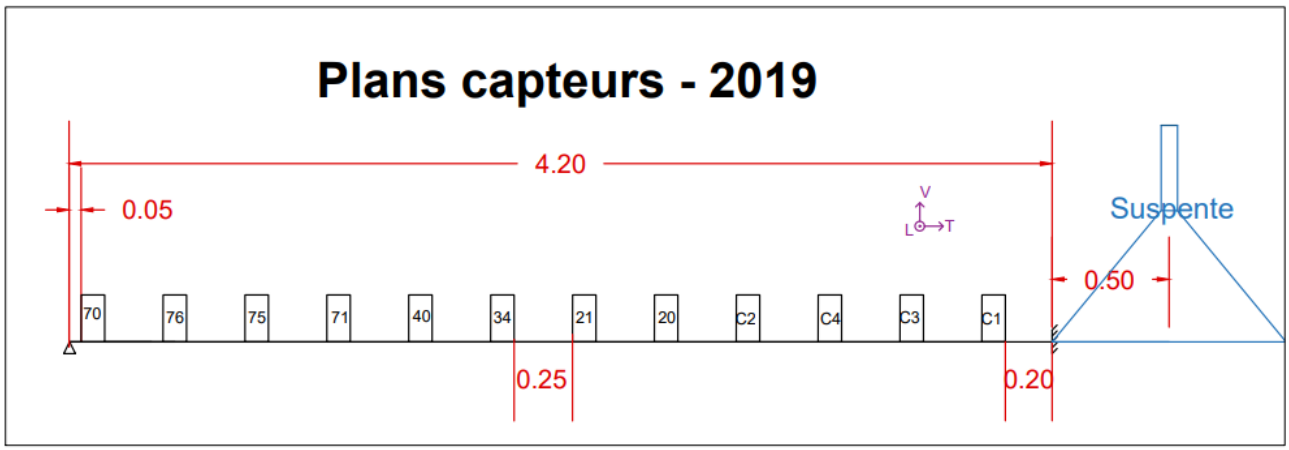

FIGURE 6.6 – Plan capteurs 2019

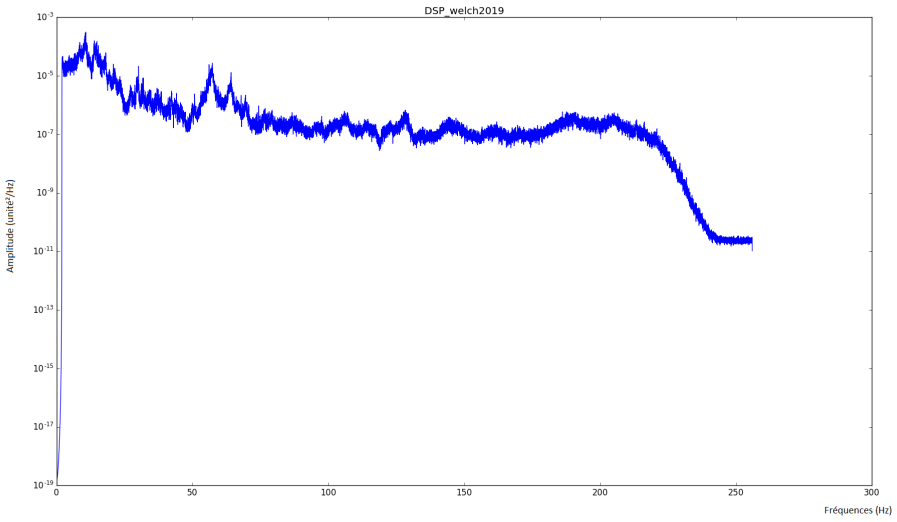

FIGURE 6.7 – Périodogramme de Welch, fenêtrage de 65536 points (2019, PM200, capteur C1, voie verticale)

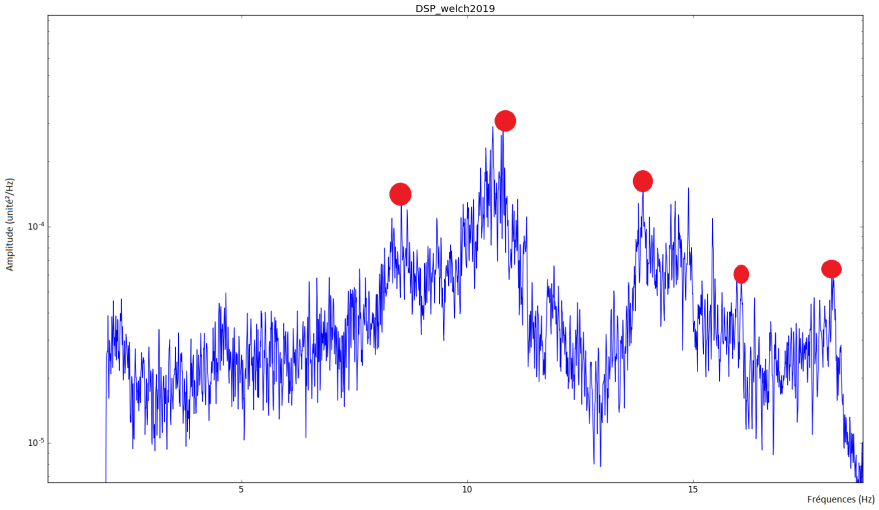

FIGURE 6.8 – Périodogramme de Welch sur les 20 premiers Hertz (2019, PM200, capteur C1, voie verticale). Les fréquences propres sont identifiées en par un point rouge.

par relevé de pics sur le périodogramme de Welch. A noter que la sixième fréquence propre par Prony est 8.39 Hertz. On reste donc dans le même ordre de grandeur que celles estimées par la première méthode.

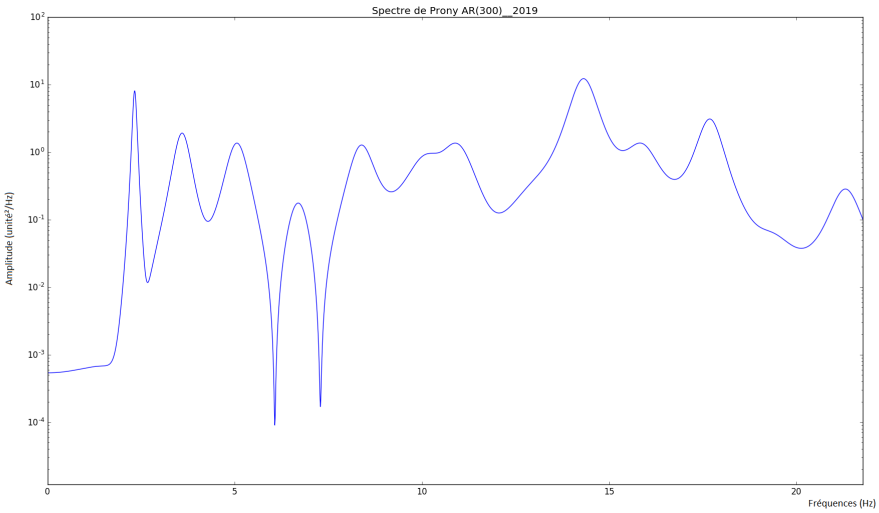

FIGURE 6.9 – Spectre de Prony d'ordre 300 (2019, PM200, Capteur C1, voie verticale)

6.2.3 Comparaison des résultats sur les 2 campagnes de mesures par rapport à un capteur

On répertorie les résultats des deux méthodes sur les capteurs 76 et C1 de 2017 et 2019 dans le tableau suivant (celles-ci sont triées par amplitudes) :

Campagne	Méthode	f_1(Hz)	f_2	f_3	f_4	f_5
2017	Welch	14.78	0.80	10.98	8.7	18.14
2017	Prony(300)	0.84	10.75	14.72	4.76	7.09
2019	Welch	10.79	13.90	8.54	18.07	16.07
2019	Prony(300)	14.32	2.33	17.69	5.06	10.89

Théoriquement ces capteurs sont écartés de trois centimètres ce qui signifie que les amplitudes des différents modes peuvent être différents. deux 'modes propres' (içi en ne s'intéressant qu'aux fréquences) semblent redondants parmi ces séries de résultats. L'une aux alentours de 14 Hertz et l'autre proche de 11 Hertz. Il est notable que ces deux fréquences se situent toujours dans ce qui a été défini comme étant des 'bons modes'. On peut donc vraisemblablement considérer que ces deux modes correspondent à des modes de la structure.

Pour ce qui est de la fréquence autour de 14 Hertz, entre 2017 et 2019 cette fréquence a chuté de 5.9 % selon la méthode de Welch et de 2.7 % selon la méthode de Prony. La fréquence autour de 11 Hertz a baissé de 1,7 % selon la méthode de Welch et a augmenté de 1,2 % selon la méthode de Prony. Une baisse de fréquence significative pourrait selon notre modèle de poutre en flexion signifier que les caractéristiques mécaniques (Module d'Young) pourraient avoir changé. Dans notre cas en s'intéressant à deux modes nous ne remarquons pas une baisse systématique pour les 2 modèles. Si on regarde plus précisément autour de 11 Hertz spectre de Prony (pour la campagne de 2019) on observe un autre pic avec une amplitude significative à 10.10 Hertz. Admettons que cette fréquence de 10.10 Hz corresponde au mode autour de 11 Hertz, dans ce cas nous n'aurions plus une augmentation mais bien une baisse de 6 % de la fréquence propre. Alors dans ce cas il y aurait deux modes propres de la structure identifiés qui auraient une baisse de fréquence. Cette baisse serait estimée entre 2.7 et 5.9 % pour le premier mode. Pour le second mode elle serait comprise entre 1.7 et 6 %. Or ces changements de fréquences ne sauraient en elles-mêmes assurer un endommagement. En effet selon Creed(1987) de telles variations (inférieurs à 5 %) ne permettent pas conclure sur un quelconque endommagement. De plus Wahab et De Roek (1997) ont démontré qu'un changement de température (de l'ordre de 15 $^\circ$C) peut entraîner une baisse de fréquence entre 4 et 5 % (cependant en tunnel la température est constante pendant toute l'année). Enfin ces changements de fréquences ne peuvent ne pas être dues seulement à un endommagement mais pourraient également être dues à des changements de conditions aux limites. Comme expliqué en introduction, nous ne sommes pas sûrs que les travaux avoisinants n'engendrent pas de phénomènes de convergences du tunnel qui pourraient encastrer la dalle de ventilation.

Cette première analyse sur une voie d'un capteur possède des limites trop importantes pour pouvoir conclure sur un quelconque endommagement. C'est pourquoi nous développons par la suite une analyse multi-capteurs.

Chapitre 7

Analyse multi-capteurs

Dans le chapitre précédent nous avons essayé d'analyser le signal selon le point de vue d'un capteur. Nous n'avons pas pu conclure sur un quelconque endommagement de la structure. De plus l'analyse sur un capteur nous prive d'une donnée importante à savoir les déformées modales. L'objectif de cette partie est d'analyser la poutre par les données de plusieurs capteurs. Tout d'abord nous allons analyser les données de la campagne 2017 selon la méthode de Prony et par frequency domain decomposition (FDD). Puis nous analyserons les données de 2019 (Nous nous intéressons seulement à la poutre située au point métrique 200).

7.1 Campagne de 2017

7.1.1 Méthode de Prony multi-capteurs

La modélisation de Prony s'applique à un seul signal. L'idée de cette méthode est de recueillir tous les modes de tous les capteurs et de retrouver les modes communs à plusieurs capteurs. De cette manière nous sommes censés obtenir les déformées de la structure en retenant les amplitudes des modes communs aux différents capteurs. Cependant en pratique les déformées obtenues ne sont pas vraisemblables. Cela peut provenir de la façon dont est résolu le système de Vandermonde pour retrouver les amplitudes. En effet les différents systèmes à résoudre diffèrent selon les capteurs et les ordres de grandeur des amplitudes ne sont donc pas forcément les mêmes. Cependant les résultats de cette méthode pour les fréquences semble assez cohérent. Nous intéressons seulement aux données des voies verticales des capteurs ainsi nous ne pourrons pas avoir accès aux modes en compression et aux modes hors plan. L'intérêt de cette méthode est d'éliminer les 'faux modes' dus au modèle de Prony.

Pour récupérer les modes communs aux différents capteurs nous procédons en plusieurs étapes. Tout d'abord on récupère tous les modes des différents capteurs dans une liste. Chaque mode correspond à une liste avec comme données le numéro du capteur, l'amplitude du mode, la phase du mode, la fréquence du mode et l'amortissement du mode. Nous possédons donc une liste de liste. Pour un ordre p nous allons récupérer p modes par capteurs (ces modes sont des modes doubles). Ainsi pour n capteurs nous allons récupérer une liste de $(n \times p)$ listes de taille cinq. Ensuite nous retirons de cette liste toute les modes qui possèdent des fréquences propres négatives (celle-ci correspondent aux conjuguéx des modes de fréquences positives). A ce stade nous avons une liste de $(n \times \frac{p}{2})$ listes de taille cinq. Nous trions ensuite cette liste par fréquence croissantes. Puis nous sélectionnons dans cette liste des sous-listes. Ces sous-listes sont sélectionnées si n sous-listes de capteurs différents se suivent dans la liste. En effet pour la campagne de 2017, cela correspond au cas où sept capteurs ont des modes avec des fréquences propres proches. Nous calculons ensuite la moyenne des fréquences de cette sous-liste. Cette sous-liste est considéré comme étant un mode de la structure si $\forall\ i\ \in [1, n],\ |f_{capteur,i} - f_{moyen,sous-liste}| \leq 0.1$. C'est dire que pour un mode les fréquences propres de tous les capteurs ne doivent pas différer à

de plus de 0.1 Hertz de la moyenne des fréquences de la sous-liste. Puis nous effectuons une nouvelle sélection sur l'amortissement de ces modes. Nous conservons les modes qui possèdent des amortissements compris entre 1 et 10 %. Nous relevons les modes pour trois ordres différents de Prony (200, 300 et 400).

Ordre 200

Ces modes sont répertoriés dans le tableau suivant :

Fréquence propre (Hertz)	Amortissement (%)	Amplitude max
8.93	8.12	0.057
14.77	2.25	0.021

On récupère donc deux modes propres qui peuvent être des modes la structure. Il se peut que ce tri soit trop sélectif et oubli certains modes propres. On le verra par la suite mais ces modes paraissent être dans le même ordre de grandeur de ceux obtenus par FDD. On rappelle également que l'amplitude correspond aux termes A_k d'un mode k dans le modèle de Prony d'ordre p d'un signal (détaillé en deuxième partie) :

$$x(n) = \sum_{k=1}^{p} A_k e^{j\phi_k} e^{n(-\alpha_k + j2\pi f_k)}$$

7.1.2 ordre 300

Fréquence propre (Hertz)	Amortissement (%)	Amplitude max
8.98	4.44	0.041
10.76	5.35	0.037
12.10	5.26	0.021
14.62	2.21	0.018
24.48	2.77	0.025

7.1.3 ordre 400

Fréquence propre (Hertz)	Amortissement (%)	Amplitude max
8.78	4.93	0.039
11.00	3.37	0.027
12.28	3.19	0.020
20.26	2.4	0.003
44.48	1.36	0.001

Plus la fréquence propre augmente plus l'amortissement diminue. Ce qui est cohérent car plus un mode possède une fréquence élevée plus son amortissement est censé être faible. De plus l'amplitude des modes à fréquence élevée est plus faible. Ces fréquences seront à comparer aux valeurs obtenues par FDD.

7.1.4 Frequency Domain Decomposition

La longueur des signaux est d'un million de points. La longueur du fenêtrage utilisé est donc de $2^{16} = 65536$ points. Comme expliqué dans la partie deux on calcule la matrice de densité spectrale puis on effectue la décomposition en valeurs singulières. Nous ne relèverons pas la valeur des amortissements par EFDD car ceux-ci paraissent faux pour les modes propres estimés(amortissements faibles). Nous effectuons comme pour l'analyse d'un capteur un 'peak

picking' des principales fréquences propres avec un pas de deux hertz (sélection d'un maximum local tous les 2 Hertz). Puis nous trions ces maximums par amplitudes décroissantes. Les fréquences propres obtenues sont 14.78, 0.73, 9.30, 11.30, 142.45 et 145.18 hertz (classée par amplitude de densité spectrale de puissance). On peut d'ores et déjà éliminer la fréquence à 0.73 Hertz qui est trop faible pour notre structure compte tenu de notre modèle analytique.

L'avantage de la FDD est qu'elle permet de récupérer les déformées pour chaque fréquence propre selon les directions L, T et V. On va donc analyser les déformées de chaque fréquence propre séparément.

Fréquence égale à 9.30 Hertz

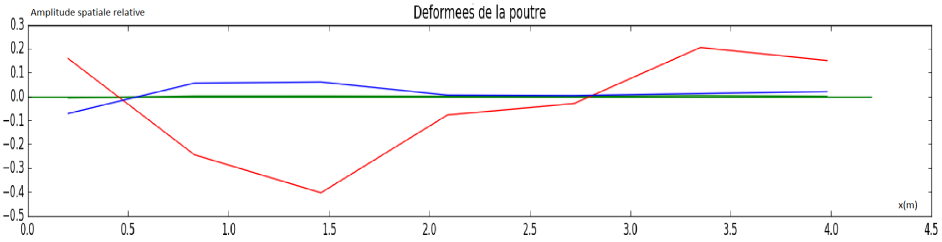

FIGURE 7.1 – Déformée obtenue par FDD à 9.3 Hertz (en rouge les déplacements verticaux, en vert les déplacements hors plan et en bleu les déplacements horizontaux). L'unité de l'amplitude spatiale relative est propre à des déplacements. Cependant comme pour la densité spectrale de puissance l'ordre de grandeur n'est pas connu.

On rappelle que la déformée correspond à la déformée d'une des deux poutres bétons constituant la structure décrite en partie 1. En rouge il s'agit du déplacement vertical, en bleu le déplacement horizontal et en vert le déplacement hors plan. Cependant le déplacement horizontal ne correspond pas à ce que l'on est censé obtenir pour le premier mode la structure. Il se peut que l'on soit proche du premier mode en fréquence. Si on se place à 8.78 Hertz (la fréquence propre obtenue par la méthode de Prony d'ordre 400) on obtient la déformée de la figure 2.1. Ce qui correspond exactement à ce l'on est censé trouver (petit déplacement horizontal négatif et grands déplacements verticaux : résultat du modèle numérique de la structure sur deux appuis simples). De plus l'amortissement de ce mode obtenu par méthode de Prony est dans le bon ordre de grandeur que l'on est censé trouver (4.93 %). Toutes ces informations démontrent que ce mode correspond au premier mode de la structure. Analytiquement ce mode avait été estimé à 7.44 hz

Fréquence de 11.30 Hertz

Première remarque ce mode est proche en fréquence d'un mode estimée par la méthode de Prony d'ordre 400. Avec la méthode de Prony nous avions trouvé une fréquence de 11 Hz soit un écart relatif de 2.5 % entre les deux méthodes. De plus la déformée ressemble fortement à ce que l'on est censé obtenir pour la déformée du deuxième mode analytiquement. De plus pour la fréquence à 11 Hz selon la méthode de Prony, le coefficient d'amortissement est de 3.37 %. Ce qui est tout à fait plausible pour une structure de génie civil. Ainsi la combinaison des informations de la méthode de Prony et de la déformée obtenue par FDD permet d'affirmer qu'il s'agit du second mode. On rappelle que analytiquement et numériquement nous avions trouvé un mode à 11.44 Hertz.

Fréquence de 14.78 Hertz

Premièrement la déformée opérationnelle correspond à la déformée du troisième mode estimée numériquement et analytiquement. Cependant contrairement aux deux autres cas nous n'avons

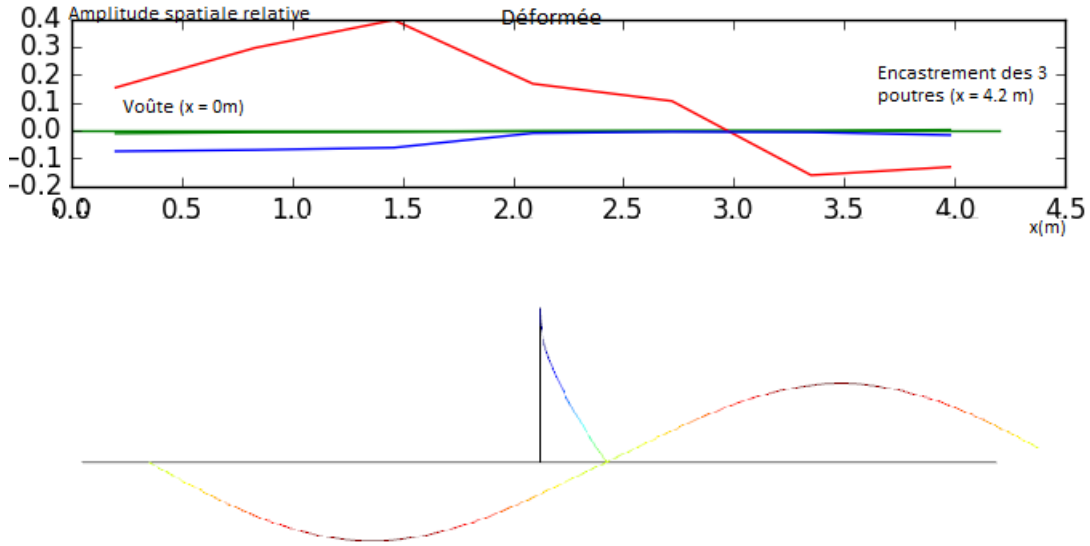

FIGURE 7.2 – (De haut en bas) Déformée obtenue par FDD à 8.78 Hertz, Déformée du premier mode obtenue numériquement

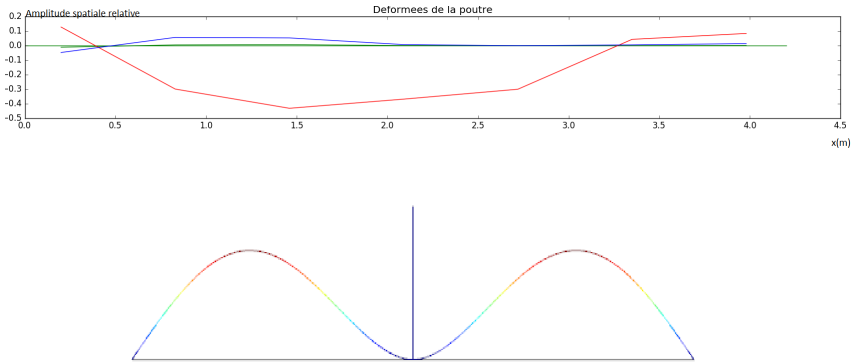

FIGURE 7.3 – (De haut en bas) Déformée obtenue par FDD à 11.3 Hertz, Déformée du deuxième mode obtenue numériquement

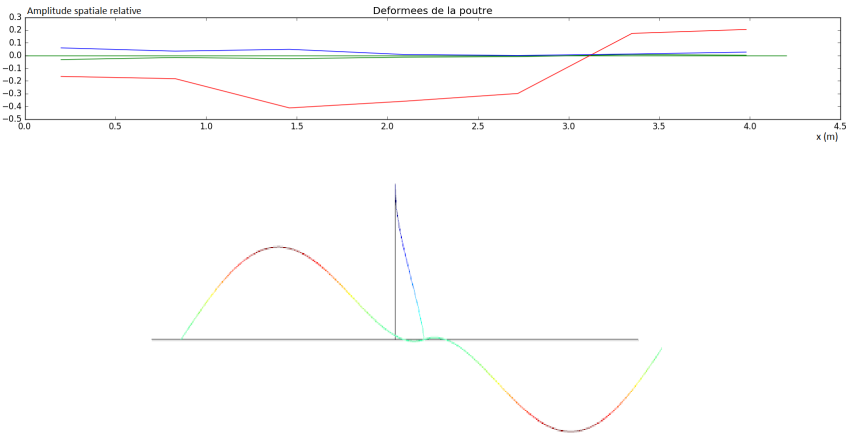

FIGURE 7.4 – (De haut en bas) Déformée obtenue par FDD à 14.78 Hertz, Déformée du troisième mode obtenue numériquement

pas directement ce mode pour le modèle de Prony d'ordre 400. Or si on augmente la tolérance en termes d'écart à la fréquence moyenne de 0.1 à 0.2 hertz pour la méthode de Prony on obtient les mêmes modes que précédemment avec (en plus) en deuxième position un mode à 14.79 hertz (avec un coefficient d'amortissement de 3.65 %). Compte-tenu du faible écart entre les 2 modèles et des informations plausibles de déformée et d'amortissement. On peut affirmer qu'il s'agit du troisième mode de la structure. On remarque d'ailleurs qu'il est situé en première position d'amplitude par FDD et en deuxième position d'amplitude par méthode de Prony. Il s'agit donc d'un mode très excité de la structure.

Fréquence de 142.45 et 145.18 Hertz

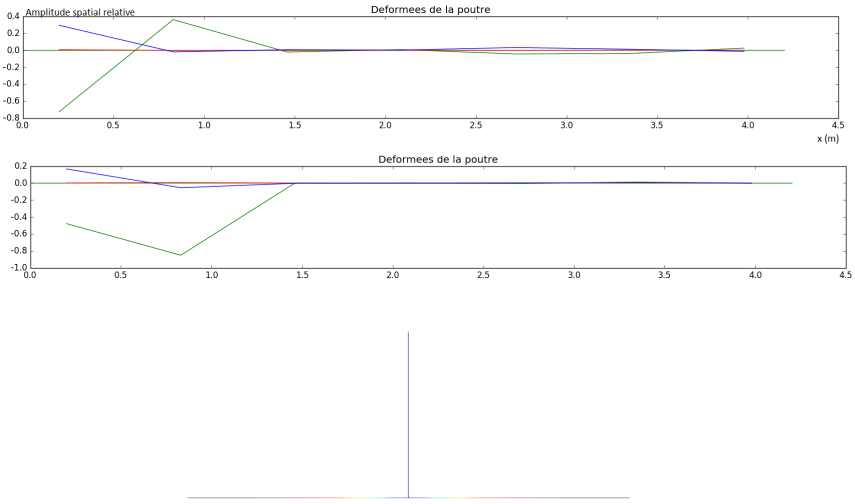

FIGURE 7.5 – (De haut en bas) Déformées obtenues par FDD à 142.45 Hertz et à 145.18 Hz, Déformée du premier mode de compression obtenue numériquement

En observant la déformée on remarque qu'il n'y a pas de déformations verticales (pas de tracé rouge). Il y a seulement du déplacement selon la direction transversale et la direction hors plan. Compte-tenu de la valeur du déplacement transversal (déplacement horizontal au nœud) de la poutre on suppose qu'il s'agit d'un mode en compression estimé analytiquement et numériquement. Nous ne le retrouvons pas par modèle de Prony puisque le modèle de Prony calcule les modes en fonction des données des voies verticales. Il n'est pas possible de trouver numériquement une solution au système de Vandermonde pour les modes horizontaux car les valeurs numériques du signal sont trop faible (peu de déplacement selon u). La déformée et l'ordre de grandeur de la fréquence propre de ce mode permettent de conclure qu'il s'agit d'un mode en compression.

7.1.5 Conclusion campagne 2017 et critiques

Les résultats en tant que tels des traitements de signaux sont difficilement interprétables. En effet à première vue nous obtenons des fréquences extrêmement faibles (autour de 0.8 hertz) qui, sans la modélisation analytique, auraient pus être conservées. La méthode de Prony permet également de discriminer ce mode à faible fréquence car il possède un amortissement trop élevé (30%). Ensuite le modèle analytique permet d'avoir un ordre de grandeur des résultats que nous sommes censés trouver pour les différents modes. Néanmoins, la méthode de Prony seule nous prive d'une information importante à savoir les déformées. Or elles sont primordiales pour la détermination des modes. La méthode de Prony constitue par contre un bon complément de vérification des modes obtenus par FDD. Tout comme la modélisation analytique de la structure permet une interprétation des résultats obtenus par FDD. Les différences entre les différentes

méthodes d'estimations des fréquences propres de trois premiers modes sont compris entre 0.001 et 5.5 %. Des résultats proches pour deux méhodes différentes sont rassurants.

Grâce à la connaissance du modèle analytique (et numérique) de la structure, des données issues de la méthode de Prony et de la FDD nous avons pu déterminer opérationnellement les trois premiers modes et le premier mode en compression de la structure en 2017. Cependant on remarque tout de même une cinématique hors plan qui ne peut être décelée que par FDD. Le modèle analytique ne prévoyait pas de tels modes du fait des hypothèses liées à celui-ci.

7.2 Campagne de 2019

A l'instar de la section 7.1, l'objectif de cette partie est dédiée à la compréhension de la dynamique de la structure en 2019. Par souci d'efficacité nous n'analyserons qu'un ordre par modèle de Prony à savoir l'ordre 400. Puis nous effectuerons un traitement du signal par FDD. Comme précédemment nous essayerons de dégager les modes principaux de la structure tout en évaluant les résultats des différents modèles. Enfin nous comparerons les résultats de l'analyse 2017 à ceux de l'analyse de 2019 de façon à conclure sur un quelconque endommagement de la structure dû aux travaux réalisés entre 2017 et 2019 dans le tunnel (creusement d'un rameau de raccordement à la galerie de sécurité).

7.2.1 Méthode de Prony d'ordre 400

Nous conservons la même méthode que celle explicitée en 7.1.1. Nous ne calculons qu'un ordre 400 car nous avions vu précédemment qu'il permettait d'avoir des résultats plus cohérent avec ceux de la FDD et également car il détectait plus de modes que des ordres moins élevés. Les modes relevés sont répertoriés dans le tableau qui suit.

Fréquence propre (Hertz)	Amortissement (%)	Amplitude max
10.82	2.78	0.018
8.66	4.43	0.012
13.74	2.79	0.012
18.12	1.85	0.008
2.93	5.20	0.008

On peut d'ores et déjà exclure la fréquence à 2.93 hertz qui est trop faible pour être une fréquence propre de la structure.

7.2.2 FDD appliqué à la campagne de 2019

On garde la même longueur de fenêtrage que précédemment. On conserve la même technique pour déterminer les fréquences propres. Les cinq premières fréquences obtenues sont 10.80, 8.54, 13.90, 18.13 et 67.87 Hertz. Les quatre premières fréquences estimées par FDD sont très proches des quatre premières fréquences estimées par méthode de Prony. Les écarts relatifs (de ces quatre premières fréquences) sont compris entre 0.06 et 1.38%. L'écart moyen est de 0.7%. On remarque de plus l'absence de modes de compression pour la FDD. Nous allons donc analyser les déformées de ces 4 premier modes.

Fréquence à 8.54 Hertz

Tout d'abord les déplacements verticaux (et horizontaux) sont quasi-nuls au niveau de la voûte du tunnel. De plus nous n'avons quasiment aucun déplacements horizontaux. Ensuite les déplacements verticaux semblent changer de signe vers la fin de la demi-poutre. Ce déplacement ressemble à la déformée du premier mode du modèle numérique de la structure lorsque l'on

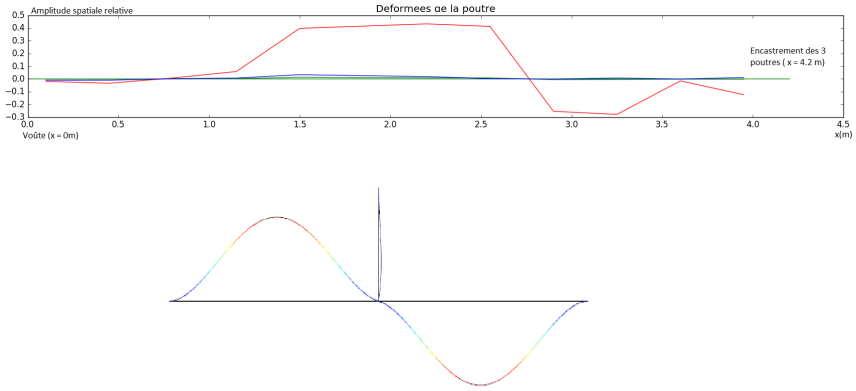

FIGURE 7.6 – (De haut en bas) Déformée obtenue par FDD à 8.54 Hertz, Déformée du premier mode obtenue numériquement (modèle à 3 encastrements)

considère qu'elle est encastrée en voûte (cinématique en $[u_M, \theta_M]$). De plus la méthode de Prony estime un coefficient d'amortissement de 4.43 % ce qui correspond à un ordre de grandeur plausible. On peut supposer que la dalle est dorénavant encastrée en voûte. L'analyse des autres modes nous en apprendra davantage.

Fréquence à 10.80 Hertz

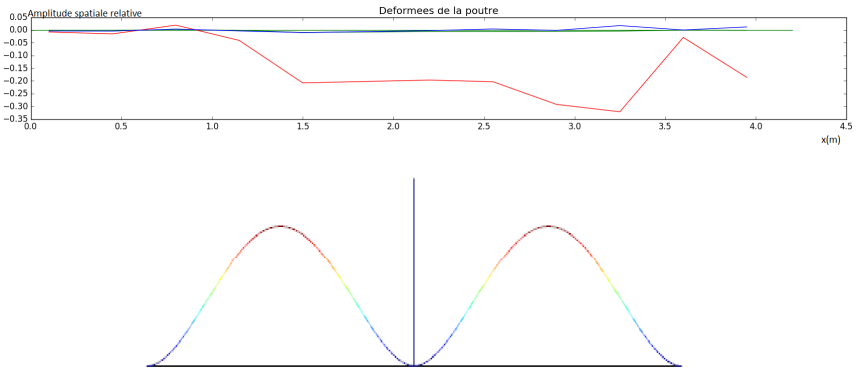

FIGURE 7.7 – (De haut en bas) Déformée obtenue par FDD à 10.8 Hertz, Déformée du deuxième mode obtenue numériquement

Les déplacements horizontaux restent quasi-nuls. Les déplacements verticaux ne changent pas de signe et sont toujours nuls en voûte. Si on ne prend pas en compte l'avant dernier point la déformée obtenue opérationnellement est quasiment la même que celle du deuxième mode du modèle de la structure à trois encastrements. Le coefficient d'amortissement du modèle de Prony est également raisonnable. Rien ne contredit le fait que la structure est dorénavant encastrée en voûte.

Fréquence à 13.90 Hertz

Il n'y a toujours pas de déplacements verticaux en voûte, pas de déplacements horizontaux, le coefficient d'amortissement est valable. Cette déformée est quasiment identique à celle du troisième mode numérique (et analytique).

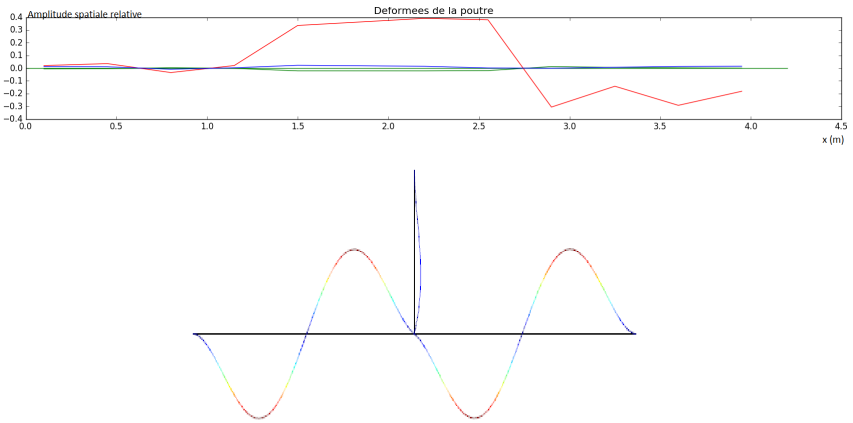

FIGURE 7.8 – (De haut en bas) Déformée obtenue par FDD à 13.9 Hertz, Déformée du troisième mode obtenue numériquement

Fréquence à 18.13 Hertz

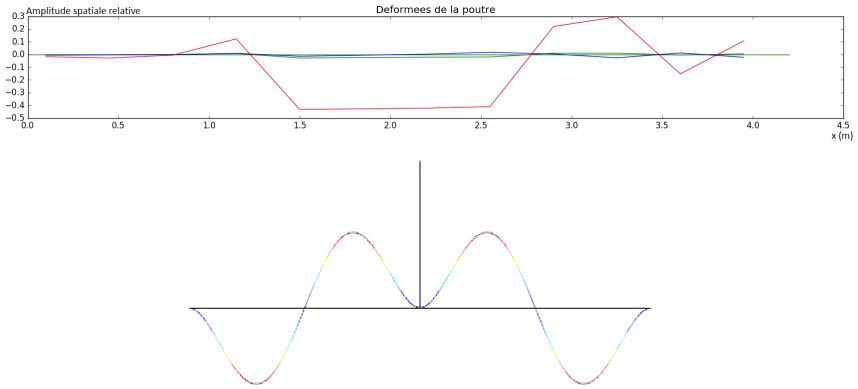

FIGURE 7.9 – (De haut en bas) Déformée obtenue par FDD à 18.13 Hertz, Déformée du quatrième mode obtenue numériquement

Il n'y a toujours pas de déplacements verticaux en voûte, pas de déplacements horizontaux, le coefficient d'amortissement est valable. Nous ne pouvons conclure sur le mode car il nous est nécessaire d'avoir la déformée de l'autre partie de la voûte pour avoir plus d'informations.

Fréquence à 67.87 Hertz

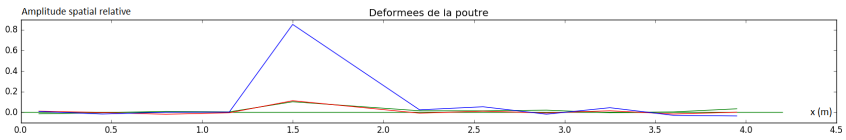

FIGURE 7.10 – Déformée obtenue par FDD à 67.87 Hertz

Il n'y aucun déplacements verticaux. Il y a des déplacements horizontaux situés près de la paroi. Cependant les déplacements horizontaux (et verticaux) sont toujours nuls en voûte. Dans notre modèle analytique à trois encastrements nous n'avons pas trouvé de mode avec ce genre de déplacements. Ce mode peut être du à un faux mode lié à la source. De plus ce mode n'est

pas détecté avec la méthode de Prony. On trouve bien numériquement un mode en compression mais à des fréquences bien plus élevées (350 Hertz).

7.2.3 Conclusion campagne 2019 et critiques

L'analyse de la campagne laisse supposer que les conditions aux limites de la structure ont changé. En 2017, il est certain que la dynamique de la structure suivait celle du modèle analytique à un encastrement et deux appuis simples. Les déformées étaient ressemblantes. Les déplacements verticaux n'étaient pas quasi-nuls près de la voûte et les déplacements horizontaux non plus. En 2019, aucun mode n'a de déplacements horizontaux ni verticaux à proximité de la voûte. De plus les déformées des premiers modes ressemblent aux déformées du modèle analytique à trois encastrements. Enfin, les modes en compression ne sont plus présent en 2019. Or ceux-ci sont caractéristiques d'un modèle à deux appuis simples. La disparition du premier mode en compression renforce l'idée selon laquelle la dynamique de la structure ne correspondrait plus à celle du modèle à deux appuis simples. On peut également penser qu'étant donné que nous avons de nouveau capteurs pour la campagne de 2019 certains nouveaux capteurs soient plus sensibles que ceux utilisés en 2017 (ce qui expliquerait des déplacements nuls près de la voûte). C'est pourquoi nous avons décidé de tester la sensibilité de ces capteurs en les laissant sur un endroit stable. L'objectif fut d'analyser l'ordre de grandeur des traces temps des capteurs. Après avoir effectué ce test, on n'observe pas de différences significatives dans l'ordre de grandeur des données recueillies par les différents capteurs.

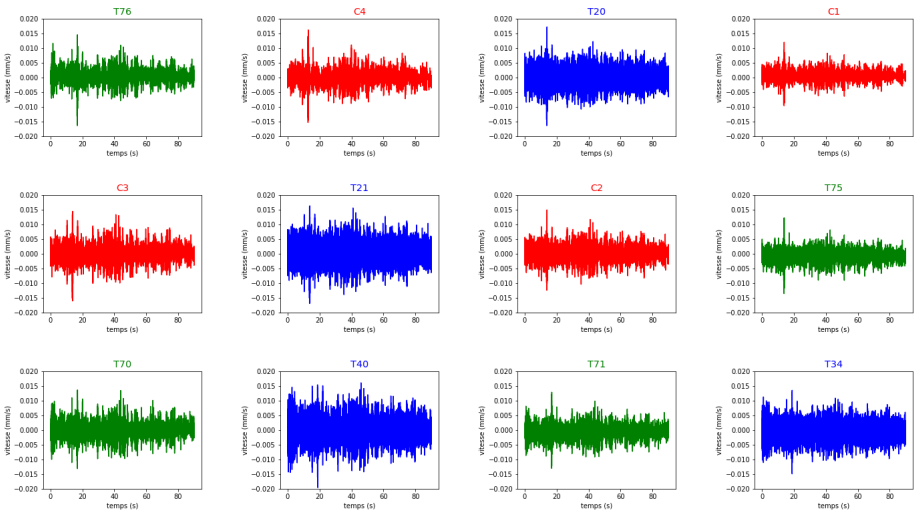

FIGURE 7.11 – Test des traces temps des différents capteurs (en bleu la première génération de capteurs, en vert la seconde et en rouge la troisième)

Le creusement du rameau qui mène à la galerie de sécurité aurait donc pu entraîner une poussée des terrains (en particulier si le rameau a été creusé à l'explosif, l'onde de choc créée par l'explosion peut créer des mouvements relatifs du massifs rocheux). Cette convergence aurait donc 'resserré' la voûte sur la dalle de ventilation. Ce resserrement aurait conduit à l'encastrement de la dalle de ventilation dans la voûte. La dynamique de la structure ne serait donc plus la même et les fréquences propres également. L'identification d'un quelconque endommagement serait impossible car le changement de fréquences propres serait dus également aux changements des conditions aux limites de la structure.

Cependant s'il s'avère qu'il n'y a pas eu de changement dans les conditions aux limites de la

dalle alors les trois premiers modes des deux campagnes de mesures sont relativement proches avec des baisses relatives des fréquences propres des trois premiers modes comprises entre 4.42 et 8.17 %. Ces baisses ne sont pas significatives d'un endommagement car elles sont de l'ordre de 5 % et l'on sait que pour de telles diminutions un endommagement n'est pas certain. On retrouve également des baisses systématiques pour le modèle de Prony d'ordre 400 sur ces trois premiers modes. Ces baisses sont comprises entre 1.4 et 7 %. Elles ne peuvent pas être dues à des différences de températures au moment des campagnes de mesures car la température reste quasiment toujours constante dans le tunnel. Enfin on a également réalisé une analyse modale par méthode de Prony d'ordre 400 et par FDD des données recueillies dans la gaine d'air vicié au point métrique 200. Les trois premiers modes sont quasiment identiques à ceux récupérés dans la gaine d'air frais. Et on ne retrouve pas par FDD la fréquence à 67.87 Hertz. Donc l'hypothèse selon laquelle ce mode correspondrait à la source lors des mesures est vérifié car ils ont été réalisé à des temps différents.

Nos résultats indiquent donc que,

- Soit la structure a changé de conditions aux limites donc la comparaison des fréquences n'est pas pertinente et l'estimation de l'endommagement n'est pas possible (première hypothèse). Les arguments qui nous poussent à supposer cela sont l'absence de modes de compression et des déplacements verticaux horizontaux (et verticaux) quasi-nuls pour tous les modes identifiés en voûte.

- Soit les conditions aux limites sont toujours les mêmes et les écarts de fréquences sont trop faibles pour être caractéristiques d'un endommagement. Les arguments qui nous poussent à supposer cela sont un encastrement trop long à la vue des déformées obtenues analytiquement et des fréquences propres qui sont relativement proches pour les deux campagnes de mesures. De plus, la compréhension du phénomène physique de poussée des terrains sur un tunnel nous est inconnu.

Conclusion

L'objectif de ce travail de fin d'études était de dégager une méthodologie d'estimation quantitative de l'endommagement par auscultation dynamique d'un ouvrage. Pour cela nous nous sommes appuyés sur un cas concret d'étude. Ce cas était la dalle de ventilation du tunnel du Siaix. Nous souhaitions savoir si la structure avait été endommagée par la réalisation de travaux avoisinants à celle-ci. Nous avons donc essayé plusieurs techniques d'analyses afin d'en dégager les avantages, les inconvénients et les limites. Ce travail est un premier pas de démarche d'inspection systématique d'ouvrage par le CETU.

Dans un premier temps, il a été primordial d'essayer d'approcher le comportement de la structure par des modèles analytiques et numériques. Même si le modèle analytique par poutre de flexion à ses limites, il permet tout de même d'avoir une référence lors de l'analyse opérationnelle de la structure. Sans une modélisation fiable il est très difficile d'analyser opérationnellement une structure. Ou du moins d'avoir un regard critique sur ceux-ci. En plus du travail sur la théorie des poutres de flexion, cette modélisation analytique a nécessité l'apprentissage d'un logiciel de calcul formel (©Maxima), de programmer des algorithmes de recherche de racines de fonctions non linéaires (©Python). La modélisation numérique a été réalisée à l'aide logiciel aux éléments finis (©Comsol).

Dans un deuxième temps, il était nécessaire d'analyser les différentes techniques d'analyses modales des structures in-situ. Notre choix a été d'opter pour l'analyse modale opérationnelle. Cette technique d'analyse a plusieurs limites. Dont la principale est certainement l'hypothèse sur le type d'excitation. En effet, en analyse modale opérationnelle on mesure la sortie (vibrations) d'une structure soumise à une excitation d'entrée. Cette excitation d'entrée est supposée être un bruit blanc. Or en réalité l'excitation n'est pas un bruit blanc. Ainsi le système que nous analysons n'est plus seulement la structure mais un système virtuel composé de la source réelle et la structure. En pratique cela amène à relever des modes propres liés à la source et plus à la structure. C'est pourquoi lors de certaines campagnes nous relevions des modes propres à fréquences extrêmement basses. Ces faux modes peuvent poser problème lors de l'analyse de la structure. Et peut importe les techniques de traitement du signal nous retrouvons ces faux modes. Nous pouvons en discriminer un certain nombre par des considérations physiques : fréquences propres trop éloignées du modèle analytique, déformées peu réaliste (on comprend l'importance d'un bon modèle analytique et numérique) et amortissement qui n'est pas dans l'ordre de grandeur des amortissements de modes de génie civil. Cependant, une incertitude sur les modes liés à la structure continue d'exister dans une certaine mesure. L'avantage de cette technique est son coût (le seul matériel nécessaire est le matériel d'acquisition des signaux) et sa faisabilité (il n'est pas nécessaire d'avoir des conditions climatiques idéales ou de bloquer la circulation).

Puis il était nécessaire de développer les différentes techniques d'analyse spectrale. Pour cela un travail préliminaire de revue d'articles scientifiques fût réalisé. Ces différents articles permettaient d'avoir un aperçu des différentes techniques actuellement utilisées dans le milieu du génie civil. Une fois l'étude théorique des ces différentes méthodes réalisée, il a été nécessaire de créer des programmes informatiques élaborés de manière à pouvoir les tester en pratique. Deux techniques de traitement du signal ont particulièrement été utilisées. Un traitement fréquentiel du signal avec la méthode de 'Frequency Domain Decomposition' (FDD) qui est une améliora-

tion des techniques fréquentielles classiques de traitement du signal comme le périodogramme, le corrélogramme et leurs versions lissées. L'autre traitement utilisé fût des modèles de Prony qui sont une amélioration des procédures auto-régressives classiques. En pratique nous nous sommes aperçus que les modèles de Prony sont difficiles à utiliser car ils nécessitent de déterminer un ordre. Cette détermination est souvent hasardeuse et il n'existe pas de techniques claires pour les déterminer. De plus, nous n'avons pas réussi à déterminer des déformées convenables à partir de ces méthodes. Cependant celles-ci permettent d'avoir un accès facile aux amortissements des différents modes. Ces techniques constituent (en termes d'analyse modale) un bon complément d'informations à la FDD. La FDD constitue une technique relativement complète d'identification modale (accès aux fréquences propres, aux déformées, aux amortissements par EFDD). Cependant l'identification des modes propres peut s'avérer délicate du fait de la très grande variation des densités spectrales de puissances estimées par cette méthode.

Enfin à partir de ces analyses il s'agissait de trancher sur l'endommagement ou non de la structure. L'endommagement est identifié en général lorsque l'on relève un changement de fréquences propres. Les fréquences propres dépendant des caractéristiques géométriques intrinsèque et des propriétés mécaniques de la structure, une baisse de fréquence signifie en général un endommagement possible de la structure. En pratique de faibles baisses de fréquences ne sont pas garantes d'un endommagement de la structure (Creed, 1987). Dans notre cas nous avons essayé de déterminer ou non l'endommagement de la dalle de ventilation du tunnel du Siaix au point mètrique 200. Or nos résultats nous amènent à penser qu'après les travaux réalisés les conditions aux limites de la structure ont changé (déformées proches d'une structure encastré au lieu de déformées de la structure de départ sur appuis simples). Puisque les fréquences propres dépendent des conditions aux limites, un changement de fréquence propre ne serait pas forcément liés à un endommagement de la structure. Si tel n'est pas le cas les écarts entre fréquences propres des deux campagnes de mesures ne sont de toute manière pas assez élevées pour conclure d'un quelconque endommagement.

Pour résumer, nous avons eu recours à une modélisation analytique (et numérique). Nous avons analysé les avantages et inconvénients des différentes techniques d'analyses modales. Nous avons pu mettre en application ces différentes techniques et en dégager les avantages et inconvénients. Enfin, un regard critique a été porté sur les résultats obtenus.

En conclusion, ce travail de fin d'études m'a permis de mettre en application une grande partie des compétences acquises lors de mon cursus d'ingénieur. Plus que des compétences, mon travail de fin d'études a été l'occasion de mettre en application une démarche d'ingénieur face à un problème (apprendre, analyser, comprendre et critiquer). Dans un démarche complète d'étude scientifique ce travail sera diffusé à travers mon rapport, ma soutenance et dans un article scientifique de l'association française de parasismique.

Bibliographie

[1] Antoine Rallu, Stéphane Hans. *Caractérisation dynamique sous bruit mécanique ambiant de structures environnant un chantier souterrain : modes transversaux du plafond de ventilation du tunnel du Siaix*

[2] Morteza Payab, Farbod Ahmadifar. *A comparative review on operational modal analysis methods.*

[3] Svend Gade, Nis B. Moller, Henrik Herlufsen, Hans Konstantin-Hansen. *Frequency domain techniques for operational modal analysis.*

[4] Lingmi Zang, Rune Brincker, Palle Andersen. *An overview of operational modal analysis : major development and issues*

[5] Camille Gontier, Clara-Serra. *Une analyse modale de type stochastique sous-espace avec prévision de l'erreur statistique sur les amortissements.*

[6] Christian Cremona, Flavio de Sousa Barbosa, Alireza Alvandi. *Identification modale sous excitation ambiante : application à la surveillance des ponts*

[7] Isaure Dunaud, Elyes El Mouelhi. *Méthode du décrément aléatoire*

[8] David Clair, Michel Folgi. *Procédures d'identification modale basées sur la méthode du décrément aléatoire et la décomposition orthogonale aux valeurs propres.*

[9] C. Cremona, F. Barbosa. *Identification modale de structures sous sollicitation ambinante : volume 1 théories et applicatons*, Laboratoire central des ponts et chaussées.

[10] A. Clement. *Détection de nouveauté pour le monitoring vibratoire des structures de génie civil : Approches chaotique et statistique de l'extraction d'indicateurs.*

[11] Bart Peeters, Guido De Roeck. *Stochastic system identification for operational modal analysis : a review*

[12] Alireza Alvendi. *Contribution à l'utilisation pratique de l'évaluation dynamique pour la détection d'endommagements dans les ponts*

[13] V.H. Vu, M. Thomas, A.A. Lakis, L.Marcouiller. *Spectrum and operational modal analysis with vector autoregressive modes.* Mechanical vibrations, nova publishers.

[14] E. Friot. *Simulation ARMA de processus stochastiques à partir de leur densité spectrale de puissance.*

[15] V.H. Vu, M. Thomas, A.A Lakis. *Operational modal analysis with time domain methods.* 24^{th} seminar on machinery vibration, Montreal.

[16] B. Peeters, M. Abdel Wahab, G. De Roeck, J. De Visscher, W.P. De Wilde, J.M. Ndambi, J. Vantomme. *Evaluation of structural damage by dynamic system identification.* Proceedings of ISMA 21, the 21^{th} international seminar on modal analysis

[17] George.H. James, Thomas.G. Carne. *The natural excitation technique (NExT) for modal parameter extraction from operating structures*. The international journal of analytical and experimental modal analysis, Janvier 1995.

[18] Olivier Besson. *Analyse spectrale paramètrique*

[19] D. Garreau, B. Georgel. *La méthode de Prony en analuse de vibrations*

[20] Creed, *Assessment of large engineering structres using data collected during in-service loading*

[21] M. A. Wahab, G. De Roeck. *Effect of temperature on dynamic system parameters of a highway bridge*. Structural engineering international.

Annexes

Matrice $\underline{\underline{K}}_1$
Du fait de sa taille importante cette matrice est présentée par colonne.
Première colonne de $\underline{\underline{K}}_1$

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{Ab\,Eb\,\chi b}{\sin(lb\,\chi b)} \\ 0 \\ \frac{Ab\,Eb\,\chi b\cos(lb\,\chi b)}{\sin(lb\,\chi b)} \end{pmatrix}$$

Deuxième colonne de $\underline{\underline{K}}_1$

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{Ab\,Eb\,\chi b}{\sin(lb\,\chi b)} \\ 0 \\ \frac{Ea\,Ia\,\beta a^2\sin(la\,\beta a)\sinh(la\,\beta a)}{\cos(la\,\beta a)\cosh(la\,\beta a)-1} \\ 0 \end{pmatrix}$$

Troisième colonne de $\underline{\underline{K}}_1$

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{Ea\,Ia\,\beta a^3\cos(la\,\beta a)\sinh(la\,\beta a)\sin(lb\,\chi b)+Ea\,Ia\,\beta a^3\cosh(la\,\beta a)\sin(la\,\beta a)\sin(lb\,\chi b)-2Ab\,Eb\cos(la\,\beta a)\cosh(la\,\beta a)\chi b\cos(lb\,\chi b)+2Ab\,Eb\,\chi b\cos(lb\,\chi b)}{(\cos(la\,\beta a)\cosh(la\,\beta a)-1)\sin(lb\,\chi b)} \\ \frac{Ea\,Ia\,\beta a^2\sin(la\,\beta a)\sinh(la\,\beta a)}{\cos(la\,\beta a)\cosh(la\,\beta a)-1} \\ \frac{Ab\,Eb\,\chi b}{\sin(lb\,\chi b)} \\ -\frac{Ab\,Eb\,\chi b}{\sin(lb\,\chi b)} \end{pmatrix}$$

Quatrième colonne de $\underline{\underline{K}}_1$

$$\begin{pmatrix} \frac{Eb\,Ib\,\beta b^2(\cosh(lb\,\beta b)-\cos(lb\,\beta b))}{\cos(lb\,\beta b)\cosh(lb\,\beta b)-1} \\ \frac{Eb\,Ib\,\beta b^2(\cosh(lb\,\beta b)-\cos(lb\,\beta b))}{\cos(lb\,\beta b)\cosh(lb\,\beta b)-1} \\ \frac{2Eb\,Ib\,\beta b^3\cos(lb\,\beta b)\sinh(lb\,\beta b)\sin(la\,\chi a)+2Eb\,Ib\,\beta b^3\cosh(lb\,\beta b)\sin(lb\,\beta b)\sin(la\,\chi a)-Aa\,Ea\cos(lb\,\beta b)\cosh(lb\,\beta b)\chi a\cos(la\,\chi a)+Aa\,Ea\,\chi a\cos(la\,\chi a)}{(\cos(lb\,\beta b)\cosh(lb\,\beta b)-1)\sin(la\,\chi a)} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Cinquième colonne de $\underline{\underline{K}}_1$

$$\begin{pmatrix} 0 \\ \frac{Eb\,Ib\,\beta b(\cos(lb\,\beta b)\sinh(lb\,\beta b)-\cosh(lb\,\beta b)\sin(lb\,\beta b))}{\cos(lb\,\beta b)\cosh(lb\,\beta b)-1} \\ -\frac{Eb\,Ib\,\beta b^2(\cosh(lb\,\beta b)-\cos(lb\,\beta b))}{\cos(lb\,\beta b)\cosh(lb\,\beta b)-1} \\ 0 \\ \frac{Eb\,Ib\,\beta b(\sinh(lb\,\beta b)-\sin(lb\,\beta b))}{\cos(lb\,\beta b)\cosh(lb\,\beta b)-1} \\ 0 \\ 0 \end{pmatrix}$$

sixième colonne de $\underline{\underline{K}}_1$

$$\begin{pmatrix} -\frac{Eb\,Ib\,\beta b\,(\cos(lb\,\beta b)\sinh(lb\,\beta b)-\cosh(lb\,\beta b)\sin(lb\,\beta b))}{\cos(lb\,\beta b)\cosh(lb\,\beta b)-1} \\ 0 \\ \frac{Eb\,Ib\,\beta b^2\,(\cosh(lb\,\beta b)-\cos(lb\,\beta b))}{\cos(lb\,\beta b)\cosh(lb\,\beta b)-1} \\ 0 \\ \frac{Eb\,Ib\,\beta b\,(\sinh(lb\,\beta b)-\sin(lb\,\beta b))}{\cos(lb\,\beta b)\cosh(lb\,\beta b)-1} \\ 0 \\ 0 \end{pmatrix}$$

septième colonne de $\underline{\underline{K}}_1$

$$\begin{pmatrix} \frac{Eb\,Ib\,\beta b\,(\sinh(lb\,\beta b)-\sin(lb\,\beta b))}{\cos(lb\,\beta b)\cosh(lb\,\beta b)-1} \\ -\frac{Eb\,Ib\,\beta b\,(\sinh(lb\,\beta b)-\sin(lb\,\beta b))}{\cos(lb\,\beta b)\cosh(lb\,\beta b)-1} \\ 0 \\ \frac{Ea\,Ia\,\beta a^2\sin(la\,\beta a)\sinh(la\,\beta a)}{\cos(la\,\beta a)\cosh(la\,\beta a)-1} \\ A \\ 0 \\ 0 \end{pmatrix}$$

Avec,

$$\begin{aligned} A = &-2Eb\,Ib\cos(la\,\beta a)\cosh(la\,\beta a)\beta b\cos(lb\,\beta b)\sinh(lb\,\beta b) - 2Eb\,Ib\,\beta b\cos(lb\,\beta b)\sinh(lb\,\beta b) \\ &-2Eb\,Ib\cos(la\,\beta a)\cosh(la\,\beta a)\beta b\cosh(lb\,\beta b)\sin(lb\,\beta b) + 2Eb\,Ib\,\beta b\cosh(lb\,\beta b)\sin(lb\,\beta b) \\ &+Ea\,Ia\,\beta a\cos(la\,\beta a)\sinh(la\,\beta a)\cos(lb\,\beta b)\cosh(lb\,\beta b) - Ea\,Ia\,\beta a\cosh(la\,\beta a)\sin(la\,\beta a)\cos(lb\,\beta b)\cosh(lb\,\beta b) \\ &-Ea\,Ia\,\beta a\cos(la\,\beta a)\sinh(la\,\beta a) + Ea\,Ia\,\beta a\cosh(la\,\beta a)\sin(la\,\beta a) \\ &/\left(\cos(la\,\beta a)\cosh(la\,\beta a)-1\right)\left(\cos(lb\,\beta b)\cosh(lb\,\beta b)-1\right) \end{aligned}$$

Liste des Programmes

Programme 7.1 – Programme permettant l'analyse modale des signaux suivant la méthode de Prony et le périodogramme de Welch (analyse par capteurs)

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jun  6 13:43:33 2019

@author: stagiaire
"""

#Modules importés-------------------------------------------------------------

#Importation du module statsmodels.tsa : analyse des séries temporelles
import statsmodels.tsa  as stm
import statsmodels.api  as sm

#Utilisation des matrices
import numpy as np
import matplotlib.pyplot as plt

#Gestion des données statistiques
import pandas as pd

#Utilisation des fonctions complexes
from cmath import *

#Traietement du signal classique
import scipy.signal as ss


#Initialisation---------------------------------------------------------------

    #Récupération des données

annee=2019 #2017 ou 2019, sinon le programme ne fonctionnera pas
PM380='non' #analyse du PM380
ordre_AR=[300] #3 ordres à traiter, ordre_AR=[a, b, c]
fech=512 #fréquence d'échantillonnage
nper=2**16 #Longueur du fenêtrage du périodogramme de Welch
K=max(ordre_AR) #taille du système de Vandermonde pour AR-Prony (au minimun égal au max(ordre_AR))

if annee==2017:
#chemin vers le fichier où sont stockées les données du signal
    chemin_signal='C:/Users/stagiaire/Documents/2017_data/decoupes/PM200/T76_0_4.dat'
    signal=pd.read_table(chemin_signal, sep='\,',header=None,index_col=None)
    signal=np.array(signal[2]) #Choix de la voie : 0=voie L, 1=voie T, 2=voie V
    signal=signal[:1000000]
    chemin_dossier_AR='C:/Users/stagiaire/Documents/AR_codes/Param AR/PM200_p_'
    Liste_chemin_dossier_AR=[chemin_dossier_AR+str(i)+'/'+'T76_p_'+str(i)+'.txt' for i in ordre_AR]
    AR=[]
    for i in Liste_chemin_dossier_AR:
        donnes=np.array(pd.read_table(i, sep='\,',header=None,index_col=None))
        V=donnes[2,:].tolist()
        V=np.array(V)
        AR.append(V)
    ARa=AR[0]
#     ARb=AR[1]
#     ARc=AR[2]

if annee==2019:
    chemin_signal='C:/Users/stagiaire/Documents/donneesPM200/PM200F/11_C1_0_1_1_m.dat'
    signal=pd.read_table(chemin_signal, sep='\,',header=None,index_col=None)
    signal=np.array(signal[2])
    signal=signal[:1000000]
    chemin_dossier_AR='C:/Users/stagiaire/Documents/AR_codes/Param AR/PM200_p_'
    Liste_chemin_dossier_AR=[chemin_dossier_AR+str(i)+'_2019/'+'11__p_'+str(i)+'.txt' for i in ordre_AR]
    AR=[]
    for i in Liste_chemin_dossier_AR:
        donnes=np.array(pd.read_table(i, sep='\,',header=None,index_col=None))
        V=donnes[2,:].tolist()
        V=np.array(V)
        AR.append(V)
    ARa=AR[0]
#     ARb=AR[1]
#     ARc=AR[2]

if PM380=='oui':
#chemin vers le fichier où sont stockées les données du signal
    chemin_signal='C:/Users/stagiaire/Documents/2017_data/decoupes/PM380/T76_0_3.dat'
    signal=pd.read_table(chemin_signal, sep='\,',header=None,index_col=None)
    signal=np.array(signal[2]) #Choix de la voie : 0=voie L, 1=voie T, 2=voie V
    signal=signal[:1000000]
    chemin_dossier_AR='C:/Users/stagiaire/Documents/AR_codes/Param AR/PM380_p_'
    Liste_chemin_dossier_AR=[chemin_dossier_AR+str(i)+'/'+'T76_p_'+str(i)+'.txt' for i in ordre_AR]
    AR=[]
    for i in Liste_chemin_dossier_AR:
        donnes=np.array(pd.read_table(i, sep='\,',header=None,index_col=None))
        V=donnes[2,:].tolist()
        V=np.array(V)
        AR.append(V)
    ARa=AR[0]
#     ARb=AR[1]
#     ARc=AR[2]

#Périodogramme de Welch-------------------------------------------------------

f, Pxx=ss.welch(signal, fs=fech, window='hanning', nperseg=nper)
fig, (DSP_welch)=plt.subplots(1,1)
DSP_welch.semilogy(f,Pxx)
```

```
DSP_welch.semilogy([f[1093],f[1382],f[1779],f[2057],f[2314]],[Pxx[1093],Pxx[1382],Pxx[1779],Pxx[2057],Pxx[2314]], 'ro')
DSP_welch.set_title('DSP_welch'+str(annee))
fig.show()

#Récupération des maximums

peaks, properties=ss.find_peaks(Pxx, distance=int(2/f[1]), height=10**(-9))
f_p=[i*f[1] for i in peaks]
f_p=zip(f_p, properties['peak_heights'])

def comp(v1, v2):
    if v1[1]<v2[1]:
        return -1
    elif v1[1]>v2[1]:
        return 1
    else :
        return 0

f_p.sort(cmp=comp)

#Traitements AR_Prony-----------------------------------------------------------------

#Calcul des racines des modèle AR
Racines_a=np.roots(np.r_[1,-ARa])**-1
#Racines_b=np.roots(np.r_[1,-ARb])**-1
#Racines_c=np.roots(np.r_[1,-ARc])**-1

#Initialisation des figures

fig_a, (Diag_ampli_a)=plt.subplots(1,1)
#fig_b, (Diag_ampli_b)=plt.subplots(1,1)
#fig_c, (Diag_ampli_c)=plt.subplots(1,1)

#Fonctions utiles
dt=1./512
def MODES1(racines, delta_t):
    MODES=[]
    for i in range(len(racines)):
        fp=phase(racines[i])/(2*np.pi*delta_t)
        #if fp>=0:
        amor=np.log(abs(racines[i]))/(2*abs(fp)*np.pi*delta_t)
        modes=[fp,amor]
        MODES.append(modes)
    return MODES

#traitement du premier ordre
dt=1./fech
Mod=MODES1(Racines_a, dt)
b=np.array(signal)[:K]
a=[]
for k in range(K):
    ak=[i**k for i in Racines_a]
    a.append(ak)
a_H=np.transpose(a)
a_H=[i.conjugate() for i in a_H]
prod=np.dot(a_H,a)
prod=np.linalg.inv(prod)
h1=np.dot(np.dot(prod,a_H),b)
print(h1)

def recup_modes_amplit(H,Modes):
#récupération des minimums
    H1=[abs(i) for i in H]
    ly=zip(H1,Modes)
    ly.sort()
    sorted_Y=[x for (x,y) in ly]
    sorted_idx=[y for (x,y) in ly]
    return (sorted_Y,sorted_idx)

ampli1,r1=recup_modes_amplit(h1,Mod)
fa=[r1[i][0] for i in range(len(r1))]
Diag_ampli_a.plot(fa, ampli1, 'o')
Diag_ampli_a.set_title('Diag_amplitude'+str(ordre_AR[0])+'__'+str(annee))
fig_a.show()

#Tracé du spectre de Prony

def S2(f, h, r, fech):
    X=0
    j=complex(0,1)
    for k in range(len(r)):
        A=1./(1-r[k]*(exp(j*2*np.pi*f/fech)**-1))
        B=1./(1-(r[k].conjugate()*exp(j*2*np.pi*f/fech))**-1)
        C=h[k]*(A-B)
        X+=C
    S=abs(X)**2
    return S

def Spectre_Prony(fmax, h, r, fech, annee, AR):
    precision=0.01
    N=int(fmax/precision)
    pas=float(fmax)/N
    S=np.zeros((N))
    F=[pas*k for k in range(N)]
    figure, (Spectre)=plt.subplots(1,1)
    for i in range(N):
        S[i]=S2(pas*i, h, r, fech)
    Spectre.semilogy(F, S)
    Spectre.set_title('Spectre de Prony AR('+str(AR)+')__'+str(annee))
```

```
    figure.show()
    return F, S


#print('calcul du spectre')
F,S=Spectre_Prony(256, h1, Racines_a, fech, annee, ordre_AR[0])
peaks1, properties1=ss.find_peaks(S, distance=int(2/F[1]), height=10**(-7))
f_p1=[i*F[1] for i in peaks1]
f_p1=zip(f_p1, properties1['peak_heights'])
f_p1.sort(cmp=comp)

#Tri bon et mauvais modes
Bons_modes=[]
for i in range(len(r1)):
    if r1[i][0]>=0 and r1[i][1]>=0.02 and r1[i][1]<=0.08:
        Bons_modes.append([r1[i], ampli1[i]])
```

Programme 7.2 – Programme qui détermine l'ordre optimal d'un signal

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 15 11:11:29 2019

@author: stagiaire
"""

#Modules importés-----------------------------------------------------------------

#Importation du module statsmodels.tsa : analyse des séries temporelles
import statsmodels.tsa  as stm
import statsmodels.api  as sm

#Utilisation des matrices
import numpy as np
import matplotlib.pyplot as plt

#Gestion des données statistiques
import pandas as pd

#Utilisation des fonctions complexes
from cmath import *

#Traitement du signal classique
import scipy.signal as ss

#Mesurer le temps d'éxécution
import time as t

#traitement de fichier
import os

##Exctraction des données---------------------------------------------------------

#Entrées
Chemin_acces='C:/Users/stagiaire/Documents/2017_data/decoupes/'
Chemin_sortie='C:/Users/stagiaire/Documents/AR_codes/Param AR/'
PM='PM741' #choix parmi PM200, PM380, PM570, PM741
pmax=20 #choix du modèle
fe=512 #fréquence d'échantillonnage

if PM=='PM200':
    Capteurs='T76_0_4.dat'
elif PM=='PM380':
    Capteurs='T76_0_3.dat'
elif PM=='PM570':
    Capteurs='T76_0_2.dat'
elif PM=='PM741':
    Capteurs='T76_0_1.dat'
else :
    print('mauvais namming')

nom_dossier='Choix ordre_'+PM+'_p_'+str(pmax)
os.mkdir(Chemin_sortie+nom_dossier)
print('%s créé' %(nom_dossier))

##Choix de l'ordre
donnes=pd.read_table(Chemin_acces+Capteurs, sep='\,', header=None, index_col=None)
#Création d'un date time index pour les séries pandas
dates_init=[i*(1./fe) for i in range(len(donnes))]
d=pd.to_datetime(dates_init, unit='s')
#Crée des object pandas avec un DateIndex
#series_L=pd.Series(np.array(donnes[0]),d)
#series_T=pd.Series(np.array(donnes[1]),d)
series_V=pd.Series(np.array(donnes[2]),d)
#Traitement des donnes
    #Création du modèle
#model_L=stm.ar_model.AR(series_L)
#model_L.initialize()
#model_T=stm.ar_model.AR(series_T)
#model_T.initialize()

model_V=stm.ar_model.AR(series_V)
model_V.initialize()


print('meilleur ordre selon aic en calcul')
start_time=t.time()
bestLag_aic=model_V.select_order(pmax,'aic','c')
print('meilleur ordre selon aic = %s' %(bestLag_aic))
temps_calcul_aic=t.time()-start_time
print('temps de calcul aic = %s' %(temps_calcul_aic))

print('meilleur ordre selon bic en calcul')
start_time=t.time()
bestLag_bic=model_V.select_order(pmax,'bic','c')
print('meilleur ordre selon bic = %s' %(bestLag_bic))
temps_calcul_bic=t.time()-start_time
print('temps de calcul bic = %s' %(temps_calcul_bic))

fichier=open(Chemin_sortie+nom_dossier+'/'+'Criteres.txt', 'a')
fichier.write('Meilleur ordre AIC = ' + str(bestLag_aic))
fichier.write('\n Temps de calcul meilleur ordre selon AIC = %s' %(temps_calcul_aic))
fichier.write('\n\n Meilleur ordre BIC = ' + str(bestLag_bic))
fichier.write('\n Temps de calcul meilleur ordre selon BIC = %s' %(temps_calcul_bic))
fichier.close()
    #fonction d'autocorrélation partielle (nlag décalage max)
```

```
print('PACF en calcul')
start_time=t.time()
plt.plot(range(pmax+1),stm.stattools.pacf(donnes[2], nlags=pmax, method='ld'), label='PACF')
plt.plot([0,pmax],[0.05,0.05], color='r')
plt.plot([0,pmax],[-0.05,-0.05], color='r')
plt.title('PACF_p_'+str(pmax))
nom_sortie='PACF_'+str(pmax)+'.png'
plt.savefig(Chemin_sortie+nom_dossier+'/'+nom_sortie)
temps_calcul_PACF=t.time()-start_time
print('temps de calcul PACF = %s' %(temps_calcul_PACF))
fichier=open(Chemin_sortie+nom_dossier+'/'+'Criteres.txt', 'a')
fichier.write('\n \n Temps de calcul PACF (p=' +str(pmax) + ') = %s' %(temps_calcul_PACF))
fichier.close()
plt.show()
```

Programme 7.3 – Programme qui teste les méthodes AR et de Prony sur le signal test

```
# -*- coding: utf-8 -*-
"""
Created on Thu May  9 14:45:46 2019

@author: stagiaire
"""
#Fonctionne
#Modules importés------------------------------------------------------------

#Signal bruité
import random

#Importation du module statsmodels.tsa : analyse des séries temporelles
import statsmodels.tsa  as stm
import statsmodels.api  as sm

#Utilisation des matrices
import numpy as np
import matplotlib.pyplot as plt

#Gestion des données statistiques
import pandas as pd

#Utilisation des fonctions complexes
from cmath import *

#Traietement du signal classique
import scipy.signal as ss
import scipy.linalg as linalg

#Création du signal artificiel ----------------------------------------------
#fonctions permetttant de créer un signal artificiel
def signal1(t,f1,f2):
    s=10*sin(2*np.pi*f1*t)+1.5*np.sin(2*np.pi*f2*t)+0.5*random.gauss(0,0.8)
    return s

def signaldiscret(fe,Np,f1,f2):
    Te=1./fe
    t=[i*Te for i in range(Np)]
    x=[signal1(i*Te,f1,f2) for i in range(Np)]
    Z=[x,t]
    return Z

#Création du signal artificiel sous forme pandas avec DateIndex
f1t=50
f2t=120
fech=512
Np=2000
Test,Temps=signaldiscret(fech,Np,f1t,f2t) #signal à traiter
d=pd.to_datetime(Temps, unit='s')
series=pd.Series(np.array(Test),d)
#     series.plot()
#     plt.show()

#traitement des données------------------------------------------------------

    #fonction d'autocorrélation partielle (nlag décalage max)
#plt.plot(range(151),stm.stattools.pacf(Test, nlags=150,method='ld'))

    #Choix de l'odre maximum
pmax=10

    #Choix du critère de sélection de l'ordre
critere='aic' #choix possibles aic, bic, hic, t-stat

    #Création du modèle
model=stm.ar_model.AR(series)
model.initialize()

    #choix du meilleur ordre pour cette série
#bestLag=model.select_order(pmax,critere,'c')
model_Fit=model.fit(maxlag=pmax, trend='nc')

    #Récupération des paramètres AR
AR=np.array(model_Fit.params)

    #Récupération des racines qui permettent de déterminer les fréquences propres
Racines_1=model_Fit.roots

#Tracé de la DSP-------------------------------------------------------------

#fonction de tracé
def traceDSPopt(paramAR,fe):
    paramAR=paramAR.tolist()
    paramAR.insert(0,1)
    time_step=1./fe
    ARfft=np.fft.fft(paramAR, n=fech)
    sample_freq=np.fft.fftfreq(len(ARfft),d=time_step)
    fplus=np.where(sample_freq>0)
    freq=sample_freq[fplus]
    puissance=np.abs(ARfft)[fplus]
    #print(puissance)
    #print(freq)
    plt.semilogy(freq,puissance)
    plt.yscale('log')
    return freq, puissance
```

```
#tracé de la dsp avec welch-------------------------------------------------------------
#tracé en fonction des AR

#f,p = traceDSPopt(AR,fech)

#f, Pxx_den=ss.welch(AR, fech, nperseg=200)
#plt.semilogy(f,Pxx_den)
#plt.xlim(0,fech/2)
#plt.show()

#Détermination des modes et amortissements-----------------------------------------

dt=1./512
def MODES(racines, delta_t):
    MODES=[]
    for i in range(len(racines)):
        fp=phase(racines[i])/(2*np.pi*dt)
        #if fp>=0:
        amor=np.log(abs(racines[i]))/(2*abs(fp)*np.pi*dt)
        modes=[fp,amor]
        MODES.append(modes)
    return MODES

Modes_p=MODES(Racines_1, dt)

###Diagramme de stabilité-----------------------------------------------------------


def diag_stab(Modele, p, delta_t):
    Modes=[]
    for k in range(1,p):
        Modele.initialize()
        Modele_Fit=model.fit(k)
        Racines=Modele_Fit.roots
        m=MODES(Racines, delta_t)
        Modes.append(m)
    for k in range(len(Modes)):
        for j in range(len(Modes[k])):
            if Modes[k][j][1]>=0 and Modes[k][j][1]<=1 :
                plt.plot([Modes[k][j][0]],[Modes[k][j][1]], 'ro')

#diag_stab(model, pmax, dt)

test_racine=np.roots(np.r_[1,-AR])**-1

K=40
b=np.array(Test)[:K]
a=np.zeros((K,len(Racines_1)), dtype=complex)
for k in range(K):
    ak=[i**k for i in Racines_1]
    a[k,:]=ak

a_H=np.transpose(a)
a_H=[i.conjugate() for i in a_H]
prod=np.dot(a_H,a)
prod=np.linalg.inv(prod)
h1=np.dot(np.dot(prod,a_H),b)

def recup_modes_amplit(H,Modes):
#récupération des minimums
    H1=[abs(i) for i in H]
    ly=zip(H1,Modes)
    ly.sort()
    sorted_Y=[x for (x,y) in ly]
    sorted_idx=[y for (x,y) in ly]
    return (sorted_Y,sorted_idx)

ampli1,r1=recup_modes_amplit(h1,Modes_p)
fa=[r1[i][0] for i in range(len(r1))]
plt.plot(fa,ampli1, 'o')


#Tracé du spectre de Prony

def S2(f, h, r, fech):
    X=0
    j=complex(0,1)
    for k in range(len(r)):
        A=1./(1-r[k]*(exp(j*2*np.pi*f/fech)**-1))
        B=1./(1-(r[k].conjugate()*exp(j*2*np.pi*f/fech))**-1)
        C=h[k]*(A-B)
        X+=C
    S=abs(X)**2
    return S

def Spectre_Prony(fmax, h, r, fech, AR):
    precision=0.01
    N=int(fmax/precision)
    pas=float(fmax)/N
    S=np.zeros((N))
    F=[pas*k for k in range(N)]
    figure, (Spectre)=plt.subplots(1,1)
    for i in range(N):
        S[i]=S2(pas*i, h, r, fech)
    Spectre.semilogy(F, S)
    Spectre.set_title('Spectre de Prony AR('+str(AR)+')__'+'Test')
    figure.show()
    return F, S
```

```
#print('calcul du spectre')
F,S=Spectre_Prony(256, h1, Racines_1, fech, pmax)
peaks1, properties1=ss.find_peaks(S, distance=int(2/F[1]), height=10**(-7))
f_p1=[i*F[1] for i in peaks1]
f_p1=zip(f_p1, properties1['peak_heights'])
f_p1.sort(cmp=comp)
```

Programme 7.4 – Programme qui teste les méthodes ARMA sur le signal test

```
# -*- coding: utf-8 -*-
"""
Created on Fri May 10 11:36:31 2019

@author: stagiaire
"""

#Modules importés-------------------------------------------------------------

#Signal bruité
import random

#Importation du module statsmodels.tsa : analyse des séries temporelles
import statsmodels.tsa  as stm
import statsmodels.api  as sm

#Utilisation des matrices
import numpy as np
import matplotlib.pyplot as plt

#Gestion des données statistiques
import pandas as pd

#Utilisation des fonctions complexes
from cmath import *

#Traietement du signal classique
import scipy.signal as ss

#Création du signal artificiel-----------------------------------------------
#fonctions permetttant de créer un signal artificiel
def signal1(t,f1,f2):
    s=10*sin(2*np.pi*f1*t)+1.5*np.sin(2*np.pi*f2*t)+0.5*random.gauss(0,0.8)
    return s

def signaldiscret(fe,Np,f1,f2):
    Te=1./fe
    t=[i*Te for i in range(Np)]
    x=[signal1(i*Te,f1,f2) for i in range(Np)]
    Z=[x,t]
    return Z

#Création du signal artificiel sous forme pandas avec DateIndex
f1t=50
f2t=120
fech=512
Np=2000
Test,Temps=signaldiscret(fech,3000,f1t,f2t) #signal à traiter
d=pd.to_datetime(Temps, unit='s')
series=pd.Series(np.array(Test),d)
    #series.plot()
    #plt.show()

##Traitement du signal--------------------------------------------------------

    #ordre du modèle ARMA(p,q)
p=6
q=2

    #modèle ARMA
model=stm.arima_model.ARMA(series,[p,q])
model_Fit=model.fit()

    #paramètre AR et MA
AR=np.array(model_Fit.params)[1:p+1]
MA=np.array(model_Fit.params)[p+1:p+q+1]

    #Racines AR
Racines=model_Fit.arroots
dt=1./512
def MODES(racines, delta_t):
    MODES=[]
    for i in range(len(racines)):
        fp=phase(racines[i])/(2*np.pi*dt)
        if fp>=0:
            amor=np.log(abs(racines[i]))/(2*np.pi*dt)
            modes=[fp,amor]
            MODES.append(modes)
    return MODES

Modes_p=MODES(Racines, dt)


#Tracé de la DSP-------------------------------------------------------------

#fonction de tracé
def traceDSPopt(paramAR,fe):
    paramAR=paramAR.tolist()
    paramAR.insert(0,1)
    time_step=1./fe
    ARfft=np.fft.fft(paramAR, n=fech)
    sample_freq=np.fft.fftfreq(len(ARfft),d=time_step)
    fplus=np.where(sample_freq>0)
    freq=sample_freq[fplus]
    puissance=np.abs(ARfft)[fplus]
    #print(puissance)
    #print(freq)
```

```
    plt.plot(freq,puissance)
    #plt.yscale('log')
    return freq

def traceDSPopt_MA(paramAR,fe):
    paramAR=paramAR.tolist()
    paramAR.insert(0,1)
    time_step=1./fe
    ARfft=np.fft.fft(paramAR, n=fech)
    sample_freq=np.fft.fftfreq(len(ARfft),d=time_step)
    fplus=np.where(sample_freq>0)
    freq=sample_freq[fplus]
    puissance=np.abs(ARfft)[fplus]
    #puissance=puissance.tolist()
    puissance=[1./k for k in puissance ]
    #print(puissance)
    #print(freq)
    plt.plot(freq,puissance)
    #plt.yscale('log')
    return freq

def traceDSP_ARMA(paramAR, paramMA, fech):
    paramAR=paramAR.tolist()
    paramAR.insert(0,1)
    paramMA=paramMA.tolist()
    paramMA.insert(0,1)
    time_step=1./fech
    ARfft=np.fft.fft(paramAR, n=fech)
    MAfft=np.fft.fft(paramMA, n=fech)
    sample_freq=np.fft.fftfreq(len(ARfft),d=time_step)
    fplus=np.where(sample_freq>0)
    freq=sample_freq[fplus]
    puissanceAR=np.abs(ARfft)[fplus]
    puissanceMA=np.abs(MAfft)[fplus]
    puissanceMA=[1./k for k in puissanceMA]
    N=len(freq)
    puissance_ARMA=[puissanceMA[k]*puissanceAR[k] for k in range(N)]
    #print(puissance)
    #print(freq)
    plt.yscale('log')
    plt.plot(freq,puissance_ARMA)

def tracesignaldirect(x,fech):
    x=x.tolist()
    time_step=1./fech
    xfft=np.fft.fft(x, n=512)
    sample_freq=np.fft.fftfreq(len(xfft),d=time_step)
    fplus=np.where(sample_freq>0)
    freq=sample_freq[fplus]
    puissance=np.abs(xfft)[fplus]
    #print(puissance)
    #print(freq)
    plt.plot(freq,puissance)
    plt.yscale('log')

#tracé en fonction des AR

traceDSP_ARMA(AR, MA, fech)
tracesignaldirect(np.array(Test), fech)
```

Programme 7.5 – Programme exploitant les données de 2017 par FDD

```
# -*- coding: utf-8 -*-
"""
Created on Fri May 17 09:00:40 2019

@author: stagiaire
"""

#Modules importés-------------------------------------------------------------

#Importation du module statsmodels.tsa : analyse des séries temporelles
import statsmodels.tsa  as stm
import statsmodels.api  as sm

#Utilisation des matrices
import numpy as np
import matplotlib.pyplot as plt

#Gestion des données statistiques
import pandas as pd

#Utilisation des fonctions complexes
from cmath import *

#Traietement du signal classique
import scipy.signal as ss
import scipy.fftpack as sf
import scipy.stats as st

##Exctraction des données-----------------------------------------------------

#Récupère les signaux enregistrées
Chemin_dossier='C:/Users/stagiaire/Documents/2017_data/decoupes/'
Liste_PM=['PM200/', 'PM380/', 'PM570/', 'PM741/']
Chemin_dossiers_sortie='C:/Users/stagiaire/Documents/AR_codes/Param AR/'
Liste_Chemin=[Chemin_dossier+i+'/' for i in Liste_PM]
Liste_Sorties=['FDD200/', 'FDD380/', 'FDD570/', 'FDD741/']
K=0
Choix_PM=Liste_Chemin[K] #Liste_Chemin[0]=PM200 etc
Chemin_sortie=Chemin_dossiers_sortie+Liste_Sorties[K]
Parametres=['T21_0_','T34_0_','T40_0_','T70_0_','T71_0_','T75_0_','T76_0_']
Parametres=[i+str(4-K)+'.dat' for i in Parametres]

##Calcul de la SVD------------------------------------------------------------
    #Initialisation

L=[]
T=[]
V=[]
Tot=[]

    #Création de la matrice des signaux
for k in Parametres:
        chemin=Choix_PM+k
        donnes=pd.read_table(chemin, sep='\,',header=None,index_col=None)
        donnes=np.array(donnes)
        L1=donnes[:,0].tolist()
        T1=donnes[:,1].tolist()
        V1=donnes[:,2].tolist()
        L1=np.array(L1)
        T1=np.array(T1)
        V1=np.array(V1)
        L.append(L1)
        T.append(T1)
        V.append(V1)
        Tot.append(L1)
        Tot.append(T1)
        Tot.append(V1)
print('calcul des matrices terminé')
Tot2=[]
for k in range(21):
    Tot2+=[Tot[k][:10**6]]


#Calcul de la SVD 'totale' (croisée des voies et de tous les capteurs)
long_voies=len(Tot2[0])
nportion=2**8
nper=2**16
CSD_tot=np.zeros((len(Tot2), len(Tot2), int(float(nper/2))+1), dtype=complex)
Noverlap=None#8
for i in range(len(Tot2)) :
    for k in range(len(Tot2)) :
        f,Pxy=ss.csd(Tot2[i], Tot2[k], fs=512., window='hann', nperseg=nper, noverlap=Noverlap)
        CSD_tot[i,k,:]=Pxy
    print('ligne %s terminée' %(i))
print('calcul CSD terminé')

s1=[]
s2=[]
modeshape=[]
for i in range(len(f)):
    u,s,v=np.linalg.svd(CSD_tot[:,:,i])
    s1.append(s[0])
    s2.append(s[1])
    modeshape+=[u[:,0]]

fig, (DSP, mac, defo, armo)=plt.subplots(4,1)
DSP.semilogy(f,s1)
```

```
DSP.semilogy(f,s2)
DSP.set_title('DSP croisee')
DSP.set_xlabel('frequences')
DSP.set_ylabel('Puissance')

#Récupération des maximums

peaks, properties=ss.find_peaks(s1, distance=int(2/f[1]), height=10**(-6))
f_p=[i*f[1] for i in peaks]
f_p=zip(f_p, properties['peak_heights'])

def comp(v1, v2):
    if v1[1]<v2[1]:
        return -1
    elif v1[1]>v2[1]:
        return 1
    else :
        return 0

f_p.sort(cmp=comp)

    #reconstitution de la déformée

defo.plot([0,4.2],[0,0], 'g')
fp=74.41
nb_per=200
sfp=int(fp/f[1]) #séléction de la fréquence du pic doit être un nombre entier
DSP.plot(f[sfp],s1[sfp],'ro')
ul=[]
uy=[]
ut=[]
x_capt=[0.2, 0.83, 1.46, 2.09, 2.72, 3.35, 3.98]
for i in range(len(x_capt)):
    ul.append(modeshape[sfp][3*i].real)
    ut.append(modeshape[sfp][3*i+1].real)
    uy.append(modeshape[sfp][3*i+2].real)
defo.plot(x_capt, ul, 'g')
defo.plot(x_capt, uy, 'r')
defo.plot(x_capt, ut, 'b')
defo.set_title('Deformees de la poutre')
defo.set_xlabel('Placement sur la poutre')
fig.show()

    #Détermination du MAC de la fréquence choisie

def L_MAC(f1):
    U1=modeshape[f1]
    U1=[k.real for k in U1]
    L1_MAC=np.zeros((len(modeshape)))
    for i in range(len(modeshape)):
        mc=modeshape[i]
        mc=[k.real for k in mc]
        num=np.vdot(U1,mc)**2
        den=np.vdot(U1,U1)*np.vdot(mc,mc)
        MAC=num/den
        L1_MAC[i]=MAC
    return L1_MAC

MAC=L_MAC(sfp)
mac.plot([0,256],[5/sqrt(33),5/sqrt(33)], 'k--' )
mac.plot(f,MAC)

    #Détermination de l'amortissement du mode


indices_moins=[sfp]
k=sfp
while s1[k-1]<=s1[k]:
    k-=1
    indices_moins.append(k)

indices_plus=[sfp]
k=sfp
while s1[k+1]<=s1[k]:
    k+=1
    indices_plus.append(k)

indices=sorted(indices_moins)
indices+=indices_plus[1:]

cloche=[0]*len(s1)
for i in indices:
    cloche[i]=s1[i]

zeroPad=1*len(cloche)
irf=np.real(sf.ifft(cloche, n=zeroPad))
nb=len(irf)
if nb%2==0:#nb pair
    moitie=nb/2
else:#nb impair
    moitie=nb/2+1

#Cloche séléctionnée
test=[s1[k] for k in indices]
f_test=[k*f[1] for k in indices]
DSP.semilogy(f_test,test, 'r')

#Dessin de la fonction d'amortissement
deltaFreqs=[f[i+1]-f[i] for i in range(len(f)-2)]
```

```
deltaFreqs=sum(deltaFreqs)/len(deltaFreqs)
fs=zeroPad*deltaFreqs
t_armo=[i/fs for i in range(moitie)]
armo.plot(t_armo,irf[:moitie])

#Repérage des zéros
t_zeros=[]
for k in range(moitie-1):
    if irf[k]==0:
        t_zeros.append(t_armo[k])
    else :
        m=irf[k+1]/irf[k]
        if m<0 :
            a=(irf[k+1]-irf[k])/(t_armo[k+1]-t_armo[k])
            zero=t_armo[k]-(irf[k]/a)
            t_zeros.append(t_armo[k])

for i in range(len(t_zeros)):
    armo.plot(t_zeros[i],0,'ro')

#Calcul de la période ajustée
Tp=1./fp
DT=[]
for i in range(nb_per*2):
    dt=t_zeros[i+1]-t_zeros[i]
    armo.plot(t_zeros[i],0,'go')
    DT.append(dt)

T_per=(2*sum(DT))/(len(DT))
f_ajust=1./T_per

#Calcul de l'amortissement
DD=[]
D_fin=[]
TT=[]
td=[]
for k in range(0, nb_per*2, 2):
    k1=int(t_zeros[k]*fs)
    k2=int(t_zeros[k+2]*fs)
    D=max(irf[k1:k2])
    x=[i for i, j in enumerate(irf) if j==D]
    DD.append(D)
    armo.plot(t_armo[x[0]],D,'bo')
    D_fin.append(np.log(D/DD[0]))
    tt=(D_fin[len(D_fin)-1]/((len(D_fin)-1)*2*np.pi))
    TT.append(tt)
    td.append(t_armo[x[0]])
amortissement=-(D_fin[len(D_fin)-1]/((len(D_fin)-1)*2*np.pi))
lr=st.linregress(td, D_fin)

armo.set_xlabel('f_ajuste=␣%s␣Hz,␣amortissement=␣%s' %(f_ajust,amortissement))

##Transformée de Fourier inverse de la première valeur singulière----------------
#ss1=s1
#for i in range(int(4/f[1])):
#    ss1[i]=0
#i_s1=sf.ifft(ss1)[:moitie]
#model=stm.ar_model.AR(i_s1)
#dt=1./fs
#fig3, (tfi)=plt.subplots(1,1)
#tfi.plot(t_armo, i_s1)
#fig3.show()
#
#def MODES(racines, delta_t):
#    MODES=[]
#    for i in range(len(racines)):
#        fp=phase(racines[i])/(2*np.pi*dt)
#        if fp>=0:
#            amor=np.log(abs(racines[i]))/(-2*fp*np.pi*dt)
#            modes=[fp,amor]
#            MODES.append(modes)
#    return MODES
#
#def diag_stab(Modele, p, delta_t):
#    fig2, (mode)=plt.subplots(1,1)
#    Modes=[]
#    for k in range(1,p):
#        Modele.initialize()
#        Modele_Fit=model.fit(k)
#        Racines=Modele_Fit.roots
#        m=MODES(Racines, delta_t)
#        Modes.append(m)
#    for k in range(len(Modes)):
#        for j in range(len(Modes[k])):
#            if Modes[k][j][1]<1 and Modes[k][j][1]>0:
#                mode.plot([Modes[k][j][0]],[Modes[k][j][1]], 'ro')
#    fig2.show()
#
#model_Fit=model.fit(maxlag=40)
#AR=model_Fit.params
#root=model_Fit.roots
#m=MODES(root,dt)
#diag_stab(model, 40, dt)
```

Programme 7.6 – Programme exploitant les données de 2019 par FDD

```
# -*- coding: utf-8 -*-
"""
Created on Mon May 27 12:12:41 2019

@author: stagiaire
"""

#Modules importés-------------------------------------------------------------

#Importation du module statsmodels.tsa : analyse des séries temporelles
import statsmodels.tsa  as stm
import statsmodels.api  as sm

#Utilisation des matrices
import numpy as np
import matplotlib.pyplot as plt

#Gestion des données statistiques
import pandas as pd

#Utilisation des fonctions complexes
from cmath import *

#Traietement du signal classique
import scipy.signal as ss
import scipy.fftpack as sf
import scipy.stats as st

##Exctraction des données-----------------------------------------------------

##Récupère les signaux enregistrées
#Air Frais
Chemin_dossier='C:/Users/stagiaire/Documents/donneesPM200/PM200F/'
Liste_PM=['01_T70_0_1_1_m.dat','02_T76_0_1_1_m.dat','03_T75_0_1_1_m.dat',
          '04_T71_0_1_1_m.dat','05_T40_0_1_1_m.dat','06_T21_0_1_1_m.dat',
          '07_T20_0_1_1_m.dat','08_C2_0_1_1_m.dat','09_C4_0_1_1_m.dat',
          '10_C3_0_1_1_m.dat','11_C1_0_1_1_m.dat']
Chemin_dossiers_sortie='C:/Users/stagiaire/Documents/FDD/FDD200_2019/'
Chemins=[Chemin_dossier+i for i in Liste_PM]

##Air Vicié
#Chemin_dossier='C:/Users/stagiaire/Documents/donneesPM200/PM200V/'
#Liste_PM=['01_T70_0_2_1_m.dat','02_T76_0_2_1_m.dat','03_T75_0_2_1_m.dat',
#'04_T71_0_2_1_m.dat','05_T40_0_2_1_m.dat','06_T21_0_2_1_m.dat',
#'07_T20_0_2_1_m.dat','08_C2_0_2_1_m.dat','09_C4_0_2_1_m.dat',
#'10_C3_0_2_1_m.dat','11_C1_0_2_1_m.dat']
#Chemin_dossiers_sortie='C:/Users/stagiaire/Documents/FDD/FDD200_2019/'
#Chemins=[Chemin_dossier+i for i in Liste_PM]


##Calcul de la SVD------------------------------------------------------------
    #Initialisation

L=[]
T=[]
V=[]
Tot=[]

    #Création de la matrice des signaux
for k in Chemins:
        chemin=k
        donnes=pd.read_table(chemin, sep='\,',header=None,index_col=None)
        donnes=np.array(donnes)
        L1=donnes[:,0].tolist()
        T1=donnes[:,1].tolist()
        V1=donnes[:,2].tolist()
        L1=np.array(L1)
        T1=np.array(T1)
        V1=np.array(V1)
        L.append(L1)
        T.append(T1)
        V.append(V1)
        Tot.append(L1)
        Tot.append(T1)
        Tot.append(V1)

#Calcul de la SVD 'totale' (croisée des voies et de tous les capteurs)
long_voies=len(Tot[0])
nportion=2**8
nper=2**16
CSD_tot=np.zeros((len(Tot), len(Tot), int(float(nper/2))+1), dtype=complex)
Noverlap=None#8
for i in range(len(Tot)) :
    for k in range(len(Tot)) :
        f,Pxy=ss.csd(Tot[i], Tot[k], fs=512., window='hann', nperseg=nper, noverlap=Noverlap)
        CSD_tot[i,k,:]=Pxy
    print('ligne %s terminée' %(i))
print('calcul CSD terminé')

s1=[]
s2=[]
modeshape=[]
for i in range(len(f)):
    u,s,v=np.linalg.svd(CSD_tot[:,:,i])
    s1.append(s[0])
    s2.append(s[1])
    modeshape+=[u[:,0]]
```

```
def freq_sommets(X, Y):
    Z=[]
    for k in range(len(Y)):
        if k==0 :
            if Y[k+1]<=Y[k]:
                Z.append(X[k])
        elif k==(len(Y)-1):
            print('fin')
        elif Y[k]>=Y[k+1] and Y[k]>=Y[k-1]:
            Z.append(X[k])
    return Z


fig, (DSP, mac, defo, armo)=plt.subplots(4,1)
DSP.semilogy(f,s1)
DSP.semilogy(f,s2)
DSP.set_title('DSP croisee')
DSP.set_xlabel('frequences')
DSP.set_ylabel('Puissance')


#Récupération des maximums

peaks, properties=ss.find_peaks(s1, distance=int(2/f[1]), height=10**(-6))
f_p=[i*f[1] for i in peaks]
f_p=zip(f_p, properties['peak_heights'])

def comp(v1, v2):
    if v1[1]<v2[1]:
        return -1
    elif v1[1]>v2[1]:
        return 1
    else :
        return 0

f_p.sort(cmp=comp)

    #reconstitution de la déformée

defo.plot([0,4.2],[0,0], 'g')
fp=77.2734375
sfp=int(fp/f[1]) #séléction de la fréquence du pic doit être un nombre entier
DSP.plot(f[sfp],s1[sfp],'ro')
ul=[]
ut=[]
uv=[]
x_capt=[0.1,0.45,0.8,1.15,1.5,2.2,2.55,2.9,3.25,3.6,3.95]
for i in range(11):
    ul.append(modeshape[sfp][3*i].real)
    ut.append(modeshape[sfp][3*i+1].real)
    uv.append(modeshape[sfp][3*i+2].real)
defo.plot(x_capt, ul, 'g')
defo.plot(x_capt, uv, 'r')
defo.plot(x_capt, ut, 'b')
defo.set_title('Deformees de la poutre')
defo.set_xlabel('Placement sur la poutre')
fig.show()

    #Détermination du MAC de la fréquence choisie

def L_MAC(f1):
    U1=modeshape[f1]
    U1=[k.real for k in U1]
    L1_MAC=np.zeros((len(modeshape)))
    for i in range(len(modeshape)):
        mc=modeshape[i]
        mc=[k.real for k in mc]
        num=np.vdot(U1,mc)**2
        den=np.vdot(U1,U1)*np.vdot(mc,mc)
        MAC=num/den
        L1_MAC[i]=MAC
    return L1_MAC

MAC=L_MAC(sfp)
mac.plot([0,256],[5/sqrt(33),5/sqrt(33)], 'k--')
mac.plot(f,MAC)

    #Détermination de l'amortissement du mode


indices_moins=[sfp]
k=sfp
while s1[k-1]<=s1[k]:
    k-=1
    indices_moins.append(k)

indices_plus=[sfp]
k=sfp
while s1[k+1]<=s1[k]:
    k+=1
    indices_plus.append(k)

indices=sorted(indices_moins)
indices+=indices_plus[1:]

cloche=[0]*len(s1)
for i in indices:
```

```
    cloche[i]=s1[i]

zeroPad=1*len(cloche)
irf=np.real(sf.ifft(cloche,n=zeroPad))
nb=len(irf)
if nb%2==0:#nb pair
    moitie=nb/2
else:#nb impair
    moitie=nb/2+1

#Cloche séléctionnée
test=[s1[k] for k in indices]
f_test=[k*f[1] for k in indices]
DSP.semilogy(f_test,test, 'r')

#Dessin de la fonction d'amortissement
deltaFreqs=[f[i+1]-f[i] for i in range(len(f)-2)]
deltaFreqs=sum(deltaFreqs)/len(deltaFreqs)
fs=zeroPad*deltaFreqs
t_armo=[i/fs for i in range(moitie)]
armo.plot(t_armo,irf[:moitie])

#Repérage des zéros
t_zeros=[]
for k in range(moitie):
    if irf[k]==0:
        t_zeros.append(t_armo[k])
    else :
        m=irf[k+1]/irf[k]
        if m<0 :
            a=(irf[k+1]-irf[k])/(t_armo[k+1]-t_armo[k])
            zero=t_armo[k]-(irf[k]/a)
            t_zeros.append(t_armo[k])

for i in range(len(t_zeros)):
    armo.plot(t_zeros[i],0,'ro')

#Calcul de la période ajustée
Tp=1./fp
nb_per=10
DT=[]
for i in range(nb_per*2):
    dt=t_zeros[i+1]-t_zeros[i]
    armo.plot(t_zeros[i],0,'go')
    DT.append(dt)

T_per=(2*sum(DT))/(len(DT))
f_ajust=1./T_per

#Calcul de l'amortissement
DD=[]
D_fin=[]
TT=[]
td=[]
for k in range(0, nb_per*2, 2):
    k1=int(t_zeros[k]*fs)
    k2=int(t_zeros[k+2]*fs)
    D=max(irf[k1:k2])
    x=[i for i, j in enumerate(irf) if j==D]
    DD.append(D)
    armo.plot(t_armo[x[0]],D,'bo')
    D_fin.append(np.log(D/DD[0]))
    tt=(D_fin[len(D_fin)-1]/((len(D_fin)-1)*2*np.pi))
    TT.append(tt)
    td.append(t_armo[x[0]])
amortissement=-(D_fin[len(D_fin)-1]/((len(D_fin)-1)*2*np.pi))
lr=st.linregress(td, D_fin)

armo.set_xlabel('f_ajuste= %s Hz, amortissement= %s' %(f_ajust,amortissement))
```

Programme 7.7 – Programme exploitant les paramètres AR de 2017 par FDD

```
# -*- coding: utf-8 -*-
"""
Created on Mon May 20 14:47:06 2019

@author: stagiaire
"""

# -*- coding: utf-8 -*-
"""
Created on Mon May 13 13:38:53 2019

@author: stagiaire
"""

#Modules importés-------------------------------------------------------------

#Importation du module statsmodels.tsa : analyse des séries temporelles
import statsmodels.tsa  as stm
import statsmodels.api  as sm

#Utilisation des matrices
import numpy as np
import matplotlib.pyplot as plt

#Gestion des données statistiques
import pandas as pd

#Utilisation des fonctions complexes
from cmath import *

#Traietement du signal classique
import scipy.signal as ss


##Exctraction des données-----------------------------------------------------

#recupère les paramètres AR calculés

ordre=40
Chemin_dossier='C:/Users/stagiaire/Documents/AR_codes/Param AR/'
Liste_PM=['PM200_p_', 'PM380_p_', 'PM570_p_', 'PM741_p_']
Liste_Chemin=[Chemin_dossier+i+str(ordre)+'/' for i in Liste_PM]
K=0
Choix_PM=Liste_Chemin[K] #Liste_Chemin[0]=PM200 etc
Parametres=['T21','T34','T40','T70','T71','T75','T76']
Parametres=[i+'_p_'+str(ordre)+'.txt' for i in Parametres]
precision_DSP=10**-2

##Calcul de la SVD------------------------------------------------------------
    #Initialisation

AR_L=[]
AR_T=[]
AR_V=[]
AR=[]

    #Création de la matrice des AR
for k in Parametres:
        chemin=Choix_PM+k
        donnes=pd.read_table(chemin, sep='\,',header=None,index_col=None)
        donnes=np.array(donnes)
        L=donnes[0,:].tolist()
        T=donnes[1,:].tolist()
        V=donnes[2,:].tolist()
        L=np.array(L)
        T=np.array(T)
        V=np.array(V)
        AR_L.append(L)
        AR_T.append(T)
        AR_V.append(V)
        AR.append(L)
        AR.append(T)
        AR.append(V)
print('calcul des matrices terminé')

    #Initialisation des graphiques
fig, (DSP_croisee, DSP_V, DSP_T, defo)=plt.subplots(4,1)

    #Calcul de la SVD 'totale' (croisée des voies et de tous les capteurs)
CSD_tot=np.zeros((len(AR), len(AR), int(512/(2*precision_DSP)+1)), dtype=complex)
long_voies=len(AR[0])
for i in range(len(AR)) :
    for k in range(len(AR)) :
        f,Pxy=ss.csd(AR[i], AR[k], fs=512., nfft=int(512/precision_DSP), noverlap=None, nperseg=len(AR[0]))
        CSD_tot[i,k,:]=Pxy

s1=[]
modeshape=[]
for i in range(len(f)):
    u,s,v=np.linalg.svd(CSD_tot[:,:,i])
    s1.append(s[0])
    modeshape+=[u[:,0]]

def freq_sommets(X, Y):
    Z=[]
    for k in range(len(Y)):
        if k==0 :
```

```
            if Y[k+1]<=Y[k]:
                Z.append(X[k])
        elif k==(len(Y)-1):
            print('fin')
        elif Y[k]>=Y[k+1] and Y[k]>=Y[k-1]:
            Z.append(X[k])
    return Z


DSP_croisee.semilogy(f[0:int(205/precision_DSP)],s1[0:int(205/precision_DSP)])
DSP_croisee.set_title('DSP croisee AR_'+str(ordre))
DSP_croisee.set_ylabel('Puissance')
f_pics=freq_sommets(f[0:int(205/precision_DSP)],s1[0:int(205/precision_DSP)])
np.savetxt(Choix_PM+'freq-pics_p_'+str(ordre)+'.txt', np.array(f_pics))

    #reconstitution de la déformée
defo.plot([0,4.2],[0,0])
sfp=int(75.11/precision_DSP) #séléction de la fréquence du pic doit être un nombre entier
uv=[]
x_capt=[0.2, 0.83, 1.46, 2.09, 2.72, 3.35, 3.98]
for i in range(7):
    uv.append(modeshape[sfp][3*i+2].real)
defo.plot(x_capt, uv)
defo.set_title('Deformees 1er mode de la poutre selon v')
defo.set_xlabel('Placement sur la poutre')



  #Calcul de la SVD sur V (croisée des voies et de tous les capteurs)
CSD_tot=np.zeros((len(AR_V), len(AR_V), int(512/(2*precision_DSP)+1)), dtype=complex)
for i in range(len(AR_V)) :
    for k in range(len(AR_V)) :
        f,Pxy=ss.csd(AR_V[i], AR_V[k], fs=512., nfft=int(512/precision_DSP), noverlap=None, nperseg=len(AR[0]))
        CSD_tot[i,k,:]=Pxy

s1=[]
modeshape=[]
for i in range(len(f)):
    u,s,v=np.linalg.svd(CSD_tot[:,:,i])
    s1.append(s[0])
    modeshape+=[u[:,0]]

DSP_V.semilogy(f[0:int(205/precision_DSP)],s1[0:int(205/precision_DSP)])
DSP_V.set_title('DSP croisee AR_V'+str(ordre))
DSP_V.set_ylabel('Puissance')
f_pics_V=freq_sommets(f[0:int(205/precision_DSP)],s1[0:int(205/precision_DSP)])
np.savetxt(Choix_PM+'freq-pics_V_p_'+str(ordre)+'.txt', np.array(f_pics))


  #Calcul de la SVD sur T (croisée des voies et de tous les capteurs)
CSD_tot=np.zeros((len(AR_T), len(AR_T), int(512/(2*precision_DSP)+1)), dtype=complex)
for i in range(len(AR_T)) :
    for k in range(len(AR_T)) :
        f,Pxy=ss.csd(AR_T[i], AR_T[k], fs=512., nfft=int(512/precision_DSP), noverlap=None, nperseg=len(AR[0]))
        CSD_tot[i,k,:]=Pxy

s1=[]
modeshape=[]
for i in range(len(f)):
    u,s,v=np.linalg.svd(CSD_tot[:,:,i])
    s1.append(s[0])
    modeshape+=[u[:,0]]

DSP_T.semilogy(f[0:int(205/precision_DSP)],s1[0:int(205/precision_DSP)])
DSP_T.set_title('DSP croisee AR_U'+str(ordre))
DSP_T.set_ylabel('Puissance')
f_pics=freq_sommets(f[0:int(205/precision_DSP)],s1[0:int(205/precision_DSP)])
np.savetxt(Choix_PM+'freq-pics_T_p_'+str(ordre)+'.txt', np.array(f_pics))
```

Programme 7.8 – Programme de traitement multi-capteurs par méthode de Prony

```
# -*- coding: utf-8 -*-
"""
Created on Mon May 20 14:47:06 2019

@author: stagiaire
"""

# -*- coding: utf-8 -*-
"""
Created on Mon May 13 13:38:53 2019

@author: stagiaire
"""

#Modules importés-------------------------------------------------------------------

#Importation du module statsmodels.tsa : analyse des séries temporelles
import statsmodels.tsa  as stm
import statsmodels.api  as sm

#Utilisation des matrices
import numpy as np
import matplotlib.pyplot as plt

#Gestion des données statistiques
import pandas as pd

#Utilisation des fonctions complexes
from cmath import *

#Traietement du signal classique
import scipy.signal as ss


##Exctraction des données-----------------------------------------------------------
#Campagne 2017
#recupère les paramètres AR calculés
ordre=400
Chemin_dossier='C:/Users/stagiaire/Documents/AR_codes/Param AR/'
Liste_PM=['PM200_p_', 'PM380_p_', 'PM570_p_', 'PM741_p_']
Liste_Chemin=[Chemin_dossier+i+str(ordre)+'/' for i in Liste_PM]
K=0
Choix_PM=Liste_Chemin[K] #Liste_Chemin[0]=PM200 etc
Parametres=['T21','T34','T40','T70','T71','T75','T76']
Parametres=[i+'_p_'+str(ordre)+'.txt' for i in Parametres]
precision_DSP=10**-2
x_capt=[0.2, 0.83, 1.46, 2.09, 2.72, 3.35, 3.98]

#récupère les chemins des signaux
Chemin_dossier_signaux='C:/Users/stagiaire/Documents/2017_data/decoupes/'
Liste_PM_signaux=['PM200/', 'PM380/', 'PM570/', 'PM741/']
Liste_Chemin_signaux=[Chemin_dossier_signaux+i for i in Liste_PM_signaux]
Choix_PM_signaux=Liste_Chemin_signaux[K] #Liste_Chemin[0]=PM200 etc
Parametres_signaux=['T21_0_','T34_0_','T40_0_','T70_0_','T71_0_','T75_0_','T76_0_']
Parametres_signaux=[i+str(4-K)+'.dat' for i in Parametres_signaux]


#Campagne de 2019
C2019='oui' #'oui' ou 'non'
if C2019=='oui':
    Liste_Chemin=[Chemin_dossier+i+str(ordre)+'_2019/' for i in Liste_PM]
    Choix_PM=Liste_Chemin[0] #Liste_Chemin[0]=PM200 etc
    Parametres=['01','02','03','04','05','06','07','08','09','10','11']
    Parametres=[i+'__p_'+str(ordre)+'.txt' for i in Parametres]
    precision_DSP=10**-2
    x_capt=[0.1,0.45,0.8,1.15,1.5,2.2,2.55,2.9,3.25,3.6,3.95]
    Choix_PM_signaux='C:/Users/stagiaire/Documents/donneesPM200/PM200F/'
    Parametres_signaux=['01_T70_0_1_1_m.dat','02_T76_0_1_1_m.dat','03_T75_0_1_1_m.dat','04_T71_0_1_1_m.dat',
                        '05_T40_0_1_1_m.dat','06_T21_0_1_1_m.dat','07_T20_0_1_1_m.dat','08_C2_0_1_1_m.dat',
                        '09_C4_0_1_1_m.dat','10_C3_0_1_1_m.dat','11_C1_0_1_1_m.dat']

#  #Air vicié
#if C2019=='oui':
#    Liste_Chemin=[Chemin_dossier+i+str(ordre)+'_2019V/' for i in Liste_PM]
#    Choix_PM=Liste_Chemin[K] #Liste_Chemin[0]=PM200 etc
#    Parametres=['01','02','03','04','05','06','07','08','09','10','11']
#    Parametres=[i+'__p_'+str(ordre)+'.txt' for i in Parametres]
#    precision_DSP=10**-2
#    x_capt=[0.1,0.45,0.8,1.15,1.5,2.2,2.55,2.9,3.25,3.6,3.95]
#    Choix_PM_signaux='C:/Users/stagiaire/Documents/donneesPM200/PM200V/'
#    Parametres_signaux=['01_T70_0_2_1_m.dat','02_T76_0_2_1_m.dat','03_T75_0_2_1_m.dat','04_T71_0_2_1_m.dat',
#    '05_T40_0_2_1_m.dat','06_T21_0_2_1_m.dat','07_T20_0_2_1_m.dat','08_C2_0_2_1_m.dat','09_C4_0_2_1_m.dat',
#    '10_C3_0_2_1_m.dat','11_C1_0_2_1_m.dat']

##Calcul de la SVD------------------------------------------------------------------
    #Initialisation

AR_L=[]
AR_T=[]
AR_V=[]
AR=[]

    #Création de la matrice des AR
for k in Parametres:
        chemin=Choix_PM+k
        donnes=pd.read_table(chemin, sep='\,',header=None,index_col=None)
        donnes=np.array(donnes)
```

```
        L=donnes[0,:].tolist()
        T=donnes[1,:].tolist()
        V=donnes[2,:].tolist()
        L=np.array(L)
        T=np.array(T)
        V=np.array(V)
        AR_L.append(L)
        AR_T.append(T)
        AR_V.append(V)
        AR.append(L)
        AR.append(T)
        AR.append(V)
print('calcul des matrices terminé')

#Diagramme de comparaison des modes-----------------------------------------------

def Racines(AR):
    R=[]
    for k in range(len(AR)):
        r=np.roots(np.r_[1,-AR[k]])**-1
        R.append(r)
    return R

racines_tot=Racines(AR)

dt=1./512
def MODES(racines, delta_t):
    MODES=[]
    for i in range(len(racines)):
        fp=phase(racines[i])/(2*np.pi*dt)
        if fp>=0:
            amor=np.log(abs(racines[i]))/(2*fp*np.pi*dt)
            modes=[fp,amor]
            MODES.append(modes)
    return MODES

#Modes_p=[]
#for k in range(len(racines_tot)):
#    m=MODES(racines_tot[k], dt)
#    Modes_p.append(m)

#amor_max=40
#for k in range(len(Modes_p)):
#        for j in range(len(Modes_p[k])):
#            if j%3==0:
#                if Modes_p[k][j][1]<amor_max:
#                    plt.plot([Modes_p[k][j][0]],[Modes_p[k][j][1]], 'ro')
#            elif j%3==1:
#                if Modes_p[k][j][1]<amor_max:
#                    plt.plot([Modes_p[k][j][0]],[Modes_p[k][j][1]], 'bo')
#            else:
#                if Modes_p[k][j][1]<amor_max:
#                    plt.plot([Modes_p[k][j][0]],[Modes_p[k][j][1]], 'go')

#test amplitude-------------------------------------------------------------------

dt=1./512
def MODES1(racines, delta_t):
    MODES=[]
    for i in range(len(racines)):
        fp=phase(racines[i])/(2*np.pi*delta_t)
        #if fp>=0:
        amor=np.log(abs(racines[i]))/(2*abs(fp)*np.pi*delta_t)
        modes=[fp,amor]
        MODES.append(modes)
    return MODES

#Récupération des modes en V
def comp(v1, v2):
    if v1[3]<v2[3]:
        return -1
    elif v1[3]>v2[3]:
        return 1
    else :
        return 0

def comp2(v1,v2):
    if v1[0]<v2[0]:
        return -1
    elif v1[0]>v2[0]:
        return 1
    else :
        return 0

def comp3(v1,v2):
    if v1[2]<v2[2]:
        return -1
    elif v1[2]>v2[2]:
        return 1
    else :
        return 0


K=ordre
MV=[]

for I in range(len(AR_V)):
    RAC=np.roots(np.r_[1,-AR_V[I]])**-1
    m1=MODES1(RAC, dt)
```

```
    #calcul des amplitudes et des phases
        #récupération du signal
    chemin_signal=Choix_PM_signaux+Parametres_signaux[I]
    signal=np.array(pd.read_table(chemin_signal,sep='\,',header=None,index_col=None))
    signal=signal[:,2]
    b=np.array(signal)[:K]
        #calcul de la pseudo-inverse
    a=[]
    for k in range(K):
        ak=[i**k for i in RAC]
        a.append(ak)
    a_H=np.transpose(a)
    a_H=[i.conjugate() for i in a_H]
    prod=np.dot(a_H,a)
    prod=np.linalg.inv(prod)
        #calcul des hk
    h1=np.dot(np.dot(prod,a_H),b)
    phase_1=[phase(i) for i in h1]
    ampli=[abs(i) for i in h1]
    for i in range(len(h1)):
        mv=[I, ampli[i], phase_1[i], m1[i][0], m1[i][1]]
        MV.append(mv)
    print('calcul ligne %s de MV terminé' %(I))
print('calcul de MV terminé')

MV.sort(cmp=comp) #tri selon la fréquence

#calcul de MV que pour des fréquences strictement positives
MV_plus=[]
for i in range(len(MV)):
    if MV[i][3]>0:
        MV_plus.append(MV[i])

#séléction des modes propres

Repertoire_modes=[]
T=range(len(AR_V))
for i in range(len(MV_plus)-len(AR_V)):
    Liste=[MV_plus[i+z][0] for z in T]
    Liste.sort()
    if Liste==T:
        L_fp=[MV_plus[i+z][3] for z in T]
        moy_f=np.mean(L_fp)
        c=0
        for k in T:
            if abs(L_fp[k]-moy_f)<=0.05: #critère de largeur du mode
                c+=1
        if c==len(AR_V):
            Repertoire_modes.append([MV_plus[i+z] for z in T])

#Test déformée premier mode
nb=0
test=Repertoire_modes[nb]
L_fp=[Repertoire_modes[nb][z][3] for z in T]
L_amor=[Repertoire_modes[nb][z][4] for z in T]
moy_f=np.mean(L_fp)
moy_amor=np.mean(L_amor)
test.sort(cmp=comp2)
y_capt=[test[i][1]*np.cos(test[i][2]) for i in range(len(test))]
plt.plot([0,4.2],[0,0], 'b')
plt.plot(x_capt, y_capt, 'r')
plt.xlabel('f_moyen= %s Hz, amortissement_moyen= %s' %(moy_f,moy_amor))
plt.show()

#Récupération des fréquences moyennes, amortissements moyens et amplitudes max

Modes_fin0=[]
for k in range(len(Repertoire_modes)):
    fp=np.mean([Repertoire_modes[k][z][3] for z in T])
    m_amor=np.mean([Repertoire_modes[k][z][4] for z in T])
    max_ampli=np.max([Repertoire_modes[k][i][1] for i in range(len(Repertoire_modes[k]))])
    Modes_fin0.append([fp, m_amor, max_ampli])

Modes_fin=[]
for k in range(len(Modes_fin0)):
    if Modes_fin0[k][1]<=0.1 and Modes_fin0[k][1]>=0.01:
        Modes_fin.append(Modes_fin0[k])

Modes_fin.sort(cmp=comp3)
print(Modes_fin)
```

Programme 7.9 – Programme testant les différentes techniques de traitement du signal sur le signal test

```
# -*- coding: utf-8 -*-
"""
Created on Thu May 23 15:59:25 2019

@author: stagiaire
"""

#Fonctionne
#Modules importés------------------------------------------------------------------

#Signal bruité
import random

#Importation du module statsmodels.tsa : analyse des séries temporelles
import statsmodels.tsa  as stm
import statsmodels.api  as sm

#Utilisation des matrices
import numpy as np
import matplotlib.pyplot as plt

#Gestion des données statistiques
import pandas as pd

#Utilisation des fonctions complexes
from cmath import *

#Traietement du signal classique
import scipy.signal as ss
import scipy.fftpack as sf

#Création du signal artificiel-----------------------------------------------------
#fonctions permetttant de créer un signal artificiel
def signal1(t,f1,f2):
    s=5*sin(2*np.pi*f1*t)+1.5*np.sin(2*np.pi*f2*t)+2*random.gauss(0,0.8)
    return s

def signaldiscret(fe,Np,f1,f2):
    Te=1./fe
    t=[i*Te for i in range(Np)]
    x=[signal1(i*Te,f1,f2) for i in range(Np)]
    Z=[x,t]
    return Z

#Création du signal artificiel sous forme pandas avec DateIndex
f1t=50
f2t=120
fech=512
Np=2000
Test,Temps=signaldiscret(fech,Np,f1t,f2t) #signal à traiter
d=pd.to_datetime(Temps, unit='s')
series=pd.Series(np.array(Test),d)
    #series.plot()
    #plt.show()

#donnes=pd.read_table('C:/Users/stagiaire/Documents/2017_data/decoupes/PM741/T21_0_1.dat',sep='\,',
#header=None,index_col=None)
#
##Création d'un date time index pour les séries pandas
fe=512
Te=1./fe
dates_init=[i*(1./fe) for i in range(len(donnes))]
d1=pd.to_datetime(dates_init, unit='s')

#Crée des object pandas avec un DateIndex
    #series_L=pd.Series(np.array(donnes[0]),d)
    #series_T=pd.Series(np.array(donnes[1]),d)
series_V=pd.Series(np.array(donnes[2]),d1)

##permet de tracer le signal directement à partir de l'objet pandas
series_V.plot()
plt.title('Signal capteur 21 2017 PM741')
plt.xlabel('Temps')
plt.ylabel('Amplitude')
plt.savefig('Signal_ capteur_21_2017_PM741.png')
plt.show()
plt.close()


##Périodogramme---------------------------------------------------------------------

f,Pxx=ss.periodogram(Test, fs=512.)
plt.plot(f,Pxx)
plt.xlim(0,256)
plt.title('Periodogramme')
plt.xlabel('frequences')
plt.ylabel('Puissance')
plt.savefig('Periodogramme.png')
plt.close()
plt.semilogy(f,Pxx)
plt.xlim(0,256)
plt.ylim(0.001,100)
plt.title('Periodogramme (ordonnees logarithmique)')
plt.xlabel('frequences')
plt.ylabel('Puissance')
plt.savefig('Periodogramme_log.png')
```

```
plt.close()

Test1=[i.real for i in Test]
#pics_periodo, dic=ss.find_peaks(Test1, height=0.1)

#Corrélogramme-----------------------------------------------------------------

correl=ss.correlate(Test, Test)
correl_plus=[abs(i) for i in correl]
tf=sf.fft(correl)
freq=sf.fftfreq(len(tf), d=1./512.)
tf_plus=[abs(i) for i in tf]
plt.plot(freq,tf_plus)
plt.xlim(0,256)
plt.title('Correlogramme')
plt.xlabel('frequences')
plt.ylabel('Puissance')
plt.savefig('Correlogramme.png')
plt.close()
plt.semilogy(freq,tf)
plt.xlim(0,256)
plt.title('Correlogramme (ordonnées logarithmique)')
plt.xlabel('frequences')
plt.ylabel('Puissance')
plt.savefig('Correlogramme_log.png')
plt.close()

##Périodogramme de Welch-------------------------------------------------------

f,Pxx=ss.welch(Test1, fs=512., window='hann', nperseg=200)
plt.plot(f,Pxx)
plt.xlim(0,256)
plt.title('Periodogramme lisse')
plt.xlabel('frequences')
plt.ylabel('Puissance')
plt.savefig('Periodogramme_Welch.png')
plt.close()

plt.semilogy(f,Pxx)
plt.xlim(0,256)
plt.title('Periodogramme lisse (ordonnées logarithmique)')
plt.xlabel('frequences')
plt.ylabel('Puissance')
plt.savefig('Periodogramme_Welch_log.png')
plt.close()

f,Pxx=ss.welch(series_V, fs=512., window='hann', nperseg=200)
plt.plot(f,Pxx)
plt.xlim(0,256)
plt.title('Periodogramme lisse capteur_21_2017_PM741')
plt.xlabel('frequences')
plt.ylabel('Puissance')
plt.savefig('Periodogramme_Welch_capteur_21_2017_PM741.png')
plt.close()

plt.semilogy(f,Pxx)
plt.xlim(0,256)
plt.title('Periodogramme lisse capteur_21_2017_PM741(logarithmique)')
plt.xlabel('frequences')
plt.ylabel('Puissance')
plt.savefig('Periodogramme_Welch_loga_capteur_21_2017_PM741.png')
plt.show()
plt.close()
```

Programme 7.10 – Programme qui calcule les paramètres AR d'ordre K de toutes les voies de tous les capteurs par poutre puis les sauvegarde dans un fichier

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 15 09:28:18 2019

@author: stagiaire
"""

#Modules importés-----------------------------------------------------------------

#Importation du module statsmodels.tsa : analyse des séries temporelles
import statsmodels.tsa  as stm
import statsmodels.api  as sm

#Utilisation des matrices
import numpy as np
import matplotlib.pyplot as plt

#Gestion des données statistiques
import pandas as pd

#Utilisation des fonctions complexes
from cmath import *

#Traitement du signal classique
import scipy.signal as ss

#Mesurer le temps d'éxécution
import time as t

#traitement de fichier
import os

##Exctraction des données---------------------------------------------------------

    #Entrées
Chemin_acces='C:/Users/stagiaire/Documents/2017_data/decoupes/'
Chemin_sortie='C:/Users/stagiaire/Documents/AR_codes/Param AR/'
PM='PM200' #choix parmi PM200, PM380, PM570, PM741
pmax=400 #choix du modèle
fe=512 #fréquence d'échantillonnage
c2019='oui' #'oui' ou 'non'


if PM=='PM200':
    Capteurs=['T76_0_4.dat','T75_0_4.dat','T71_0_4.dat','T70_0_4.dat','T40_0_4.dat','T21_0_4.dat','T34_0_4.dat']
elif PM=='PM380':
    Capteurs=['T76_0_3.dat','T75_0_3.dat','T71_0_3.dat','T70_0_3.dat','T40_0_3.dat','T21_0_3.dat','T34_0_3.dat']
elif PM=='PM570':
    Capteurs=['T76_0_2.dat','T75_0_2.dat','T71_0_2.dat','T70_0_2.dat','T40_0_2.dat','T21_0_2.dat','T34_0_2.dat']
elif PM=='PM741':
    Capteurs=['T76_0_1.dat','T75_0_1.dat','T71_0_1.dat','T70_0_1.dat','T40_0_1.dat','T21_0_1.dat','T34_0_1.dat']
else :
    print('mauvais namming')

nom_dossier=PM+'_p_'+str(pmax)

#Campagne 2019--PM200F--
if c2019=='oui':
    Capteurs=['01_T70_0_1_1_m.dat','02_T76_0_1_1_m.dat','03_T75_0_1_1_m.dat','04_T71_0_1_1_m.dat',
              '05_T40_0_1_1_m.dat','06_T21_0_1_1_m.dat','07_T20_0_1_1_m.dat','08_C2_0_1_1_m.dat',
              '09_C4_0_1_1_m.dat','10_C3_0_1_1_m.dat','11_C1_0_1_1_m.dat']
    Chemin_acces='C:/Users/stagiaire/Documents/donneesPM200/PM200F/'
    Chemin_sortie='C:/Users/stagiaire/Documents/AR_codes/Param AR/'
    nom_dossier=PM+'_p_'+str(pmax)+'_'+'2019'

##Air vicié
#if c2019=='oui':
#    Capteurs=['01_T70_0_2_1_m.dat','02_T76_0_2_1_m.dat','03_T75_0_2_1_m.dat','04_T71_0_2_1_m.dat',
#    '05_T40_0_2_1_m.dat','06_T21_0_2_1_m.dat','07_T20_0_2_1_m.dat','08_C2_0_2_1_m.dat','09_C4_0_2_1_m.dat',
#    '10_C3_0_2_1_m.dat','11_C1_0_2_1_m.dat']
#    Chemin_acces='C:/Users/stagiaire/Documents/donneesPM200/PM200V/'
#    Chemin_sortie='C:/Users/stagiaire/Documents/AR_codes/Param AR/'
#    nom_dossier=PM+'_p_'+str(pmax)+'_'+'2019V'

os.mkdir(Chemin_sortie+nom_dossier)
print('%s créé' %(nom_dossier))

#Récupération
for i in Capteurs:
    print(i + 'en cours de traitement')
    start_time=t.time()
    Capteur=i
    donnes=pd.read_table(Chemin_acces+i, sep='\,', header=None, index_col=None)
    #Création d'un date time index pour les séries pandas
    dates_init=[i*(1./fe) for i in range(len(donnes))]
    d=pd.to_datetime(dates_init, unit='s')
    #Crée des object pandas avec un DateIndex
    series_L=pd.Series(np.array(donnes[0]),d)
    series_T=pd.Series(np.array(donnes[1]),d)
    series_V=pd.Series(np.array(donnes[2]),d)
    #Traitement des donnes
        #Création du modèle
    model_L=stm.ar_model.AR(series_L)
    model_L.initialize()
    model_T=stm.ar_model.AR(series_T)
    model_T.initialize()
```

```
model_V=stm.ar_model.AR(series_V)
model_V.initialize()
model_L_Fit=model_L.fit(maxlag=pmax)
model_T_Fit=model_T.fit(maxlag=pmax)
model_V_Fit=model_V.fit(maxlag=pmax)
AR_L=np.array(model_L_Fit.params)[1:]
AR_T=np.array(model_T_Fit.params)[1:]
AR_V=np.array(model_V_Fit.params)[1:]
print('Ordre AR_L : %s ; Ordre AR_T : %s ; Ordre AR_V : %s' %(len(AR_L), len(AR_T), len(AR_V)))
AR_LTV=[AR_L,AR_T,AR_V]
    #Sauvegarde des fichiers dans un dossier
nom_sortie=Capteur[0:3]+'_p_'+str(pmax)+'.txt'
np.savetxt(Chemin_sortie+nom_dossier+'/'+nom_sortie, AR_LTV, delimiter=',')
    #Temps de traitement
print("temps de traitement %s : %s secondes ---" %(Capteur[0:3], t.time()-start_time))
```

Programme 7.11 – Programme qui calcule les paramètres AR d'ordre K par l'algorithme de Levinson

```
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 12 15:23:21 2019

@author: stagiaire
"""

import math
from matplotlib import *
from pylab import *
import random
import numpy as np
from cmath import *
from pretraitement import Pretraitement

def signal(t):
    s=0.9*math.sin(2*math.pi*50*t)+0.5*math.sin(2*math.pi*15.6*t)+1.5*math.sin(2*math.pi*30*t)
#+0.3*random.gauss(0,1)+0.6*random.gauss(0,1)
    s=np.sin(2.*np.pi*10.*t)
    return s


def courbesignal(T,Np):
    dt=T/Np
    print(dt)
    t=numpy.zeros(Np)
    x=numpy.zeros(Np)
    for k in range(0,Np):
        t[k]=dt*k
        x[k]=signal(t[k])
    plot(t,x)
    return dt

def signaldiscret(T,Np):
    dt=T/Np
    print dt
    t=[i*dt for i in range(Np)]
    x=[signal(i*dt) for i in range(Np)]
    Z=[x,t]
#    plot(t,x)
    return Z

TEST=signaldiscret(1.0,10**3)
XTEST=TEST[0]
fech=10**3

#Calcul le coefficient d'autocorrélation d'un ensemble de données à l'ordre p
def autocorrel(Z1,p):
    C=0
    L=len(Z1)
    if p>=0:
        for k in range(p,L):
            C+=Z1[k]*Z1[k-p]
        C=C/(L-p)
    else:
        for k in range(-1*p,L):
            C+=Z1[k]*Z1[k+p]
        C=C/(L+p)
    return C

def autocorrel2(Z1):
    N=len(Z1)
    y=correlate(Z1,Z1,mode='full')
    c=y/N
    return c
#    tau=arange(-N,N-1,dtype=float)
#    plot(tau,c)
#    print(len(c))

#Permet de récupérer la matrice d'autocorrélation d'ordre p d'un modèle AR
def matrice_autocorrel(Z1,p):
    M=numpy.zeros((p+1,p+1))
    R=numpy.zeros(2*p+1)
    for i in range(0,p+1):
        R[i]=autocorrel(Z1,i)
        R[i]=autocorrel2(Z1)[i]
    for i in range(1,p+1):
        R[i+p]=autocorrel(Z1,-i)
        R[i+p]=autocorrel2(Z1)[i+p]
    for k in range(0,p+1):
        for i in range(0,p+1):
            if i-k>=0:
                M[i,k]=R[i-k]
            else :
                M[i,k]=R[p+k-i]
    return M

def trace_ac(Z1,p):
    R=numpy.zeros(p+1)
    X=range(p+1)
    for k in range(p+1):
        R[k]=autocorrel(Z1,k)
    plot(X,R)


##Permet d'obtenir tous les paramètres des modèles auto-régressif d'ordre inférieurs à p
```

```
def algo_Levinson(Z1,p):
  #phase d'initialisation
    M=matrice_autocorrel(Z1,p)
    A=numpy.zeros((p+1,p))
    rho=numpy.zeros(p)
    A[1,0]=-M[1,0]/M[0,0]
    rho[0]=(1-abs(A[1,0])**2)*M[0,0]
    for j in range(len(A[0])):
        A[0,j]=1
    #Phase de récurrence
    for j in range(2,p+1):
        A[j,j-1]=-M[j,0]/rho[j-2]
        for i in range(1,j):
            A[j,j-1]-=A[i,j-2]*M[j,i]/rho[j-2]
        for i in range(1,j):
            A[i,j-1]=A[i,j-2]+A[j,j-1]*A[j-i,j-2]
        rho[j-1]=(1-abs(A[j,j-1])**2)*rho[j-2]
    return A

#A partir de la matrice des paramètres de modèle AR permet d'obtenir la densité spectrale de puissance à
la fréquence f
def all_DSP(A,f):
    p=len(A)
    DSP=numpy.zeros((p-1,1),dtype=complex)
    DSP1=numpy.zeros((p-1,1))
    z=complex(0,0)
    z1=complex(0,0)
    for i in range(1,p-1):
        for k in range(0,i):
            z=complex(0,-2*pi*k*f/fech)
            z1=exp(z)
            DSP[i,0]=DSP[i,0]+complex(A[k,i],0)*z1
        DSP1[i,0]=1/(abs(DSP[i,0])**2)
    DSP1=delete(DSP1,0,axis=0)
    return DSP1

def DSP(A,f):
    p=len(A)-1
    DSP=complex(0,0)
    z=complex(0,0)
    for k in range(0,p+1):
        z=complex(0,-2*pi*k*f/fech)
        z1=exp(z)
        a=complex(A[k,p-1],0)
        DSP+=a*z1
    DSP=1./(abs(DSP)**2)
    return DSP

#Calcul l'ensembe des valeurs de la DSP pour des paramètres inf ou eg à p pour f inf ou eg à fmax avec un pas
def TDSP(Z1,p,fmax,pas):
    A=algo_Levinson(Z1,p+2)
    N=int(ceil(fmax/pas))
    TDSP=numpy.zeros((p,N))
    for k in range(0,N):
        f=k*pas
        TDSP[:,k]=DSP(A,f).reshape(p)
    return TDSP

#trace la dsp d'un modèle AR d'ordre p
def trace_all_dsp(Z1,p,fmax,pas):
    tdsp=TDSP(Z1,p,fmax,pas)
    N=int(ceil(fmax/pas))
    F=numpy.zeros(N)
    for k in range(0,N):
        F[k]=k*pas
    X=F
    for k in range(0,p):
        Y=tdsp[k,:]
        plot(X,Y)

def trace_DSP(A,fmax,pas):
    N=int(ceil(fmax/pas))
    F=numpy.zeros(N)
    S=numpy.zeros(N)
    for k in range(0,N):
        f=k*pas
        S[k]=DSP(A,f)
        F[k]=f
    print([S[i] for i in range(10)])
    plot(F,S)
```

Programme 7.12 – Programme qui détermine les zéros des fonctions analytiques de la partie 1

```
# -*- coding: utf-8 -*-
"""
Created on Mon May  6 14:45:39 2019

@author: stagiaire
"""

import math
from matplotlib import *
from pylab import *
import random
import numpy as np
import scipy


##fonctions dont les zéros sont à rechercher

def g1(f):
    g1=((125149.2246406247*abs(f)**1.5*cos(1.160858784392945*sqrt(abs(f)))**2*sinh(1.160858784392945*sqrt(abs(f)))**2
         -125149.2246406247*abs(f)**1.5*cosh(1.160858784392945*sqrt(abs(f)))**2*sin(1.160858784392945*sqrt(abs(f)))**2
         +125149.2246406247*abs(f)**1.5*cosh(1.160858784392945*sqrt(abs(f)))**2-250298.4492812495*abs(f)**1.5
         *cos(1.160858784392945*sqrt(abs(f)))*cosh(1.160858784392945*sqrt(abs(f)))
         +125149.2246406247*abs(f)**1.5*cos(1.160858784392945*sqrt(abs(f)))**2)*sin(0.003890567728186338*abs(f))
        +0.001897837916188458*abs(f)*((1.5168*10**10*cos(1.160858784392945*sqrt(abs(f)))
        -1.5168*10**10*cos(1.160858784392945*sqrt(abs(f)))**2*cosh(1.160858784392945*sqrt(abs(f))))
        *sinh(1.160858784392945*sqrt(abs(f)))+(1.5168*10**10*cos(1.160858784392945*sqrt(abs(f)))
        *cosh(1.160858784392945*sqrt(abs(f)))**2
        -1.5168*10**10*cosh(1.160858784392945*sqrt(abs(f))))*sin(1.160858784392945*sqrt(abs(f))))
        *cos(0.003890567728186338*abs(f)))
        /((cos(1.160858784392945*sqrt(abs(f)))**2*cosh(1.160858784392945*sqrt(abs(f)))
        -cos(1.160858784392945*sqrt(abs(f))))*sinh(1.160858784392945*sqrt(abs(f)))
        +(cosh(1.160858784392945*sqrt(abs(f)))-cos(1.160858784392945*sqrt(abs(f)))
        *cosh(1.160858784392945*sqrt(abs(f)))**2)*sin(1.160858784392945*sqrt(abs(f))))*sin(0.003890567728186338*abs(f)))
    return g1

def g2(f):
    g2=-(((0.2321717568785891*sqrt(abs(f))*(10000000.0*cos(0.4183880111235364*sqrt(abs(f)))
    *cosh(0.4183880111235364*sqrt(abs(f)))-10000000.0)*cos(1.160858784392945*sqrt(abs(f)))**2
    +0.2321717568785891*sqrt(abs(f))*(10000000.0-10000000.0*cos(0.4183880111235364*sqrt(abs(f)))
    *cosh(0.4183880111235364*sqrt(abs(f))))*sinh(1.160858784392945*sqrt(abs(f)))**2
    +((0.2321717568785891*sqrt(abs(f))*(2.0*10**7-2.0*10**7*cos(0.4183880111235364*sqrt(abs(f)))
    *cosh(0.4183880111235364*sqrt(abs(f))))*cos(1.160858784392945*sqrt(abs(f)))*cosh(1.160858784392945*sqrt(abs(f)))
    +0.2321717568785891*sqrt(abs(f))*(2.0*10**7*cos(0.4183880111235364*sqrt(abs(f)))
    *cosh(0.4183880111235364*sqrt(abs(f)))-2.0*10**7))*sin(1.160858784392945*sqrt(abs(f)))
    +(6426439.85085752*sqrt(abs(f))*cos(0.4183880111235364*sqrt(abs(f)))*sinh(0.4183880111235364*sqrt(abs(f)))
    -6426439.85085752*sqrt(abs(f))*cosh(0.4183880111235364*sqrt(abs(f)))*sin(0.4183880111235364*sqrt(abs(f))))
    *cos(1.160858784392945*sqrt(abs(f)))**2*cosh(1.160858784392945*sqrt(abs(f)))+(6426439.85085752*sqrt(abs(f))
    *cosh(0.4183880111235364*sqrt(abs(f)))*sin(0.4183880111235364*sqrt(abs(f)))-6426439.85085752*sqrt(abs(f))
    *cos(0.4183880111235364*sqrt(abs(f)))*sinh(0.4183880111235364*sqrt(abs(f))))*cos(1.160858784392945*sqrt(abs(f))))
    *sinh(1.160858784392945*sqrt(abs(f)))+(0.2321717568785891*sqrt(abs(f))
    *(10000000.0*cos(0.4183880111235364*sqrt(abs(f)))*cosh(0.4183880111235364*sqrt(abs(f)))-10000000.0)
    *cosh(1.160858784392945*sqrt(abs(f)))**2+0.2321717568785891*sqrt(abs(f))*(10000000.0-10000000.0
          *cos(0.4183880111235364*sqrt(abs(f)))*cosh(0.4183880111235364*sqrt(abs(f))))
    *sin(1.160858784392945*sqrt(abs(f)))**2+((6426439.85085752*sqrt(abs(f))*cosh(0.4183880111235364*sqrt(abs(f)))
    *sin(0.4183880111235364*sqrt(abs(f)))-6426439.85085752*sqrt(abs(f))*cos(0.4183880111235364*sqrt(abs(f)))
    *sinh(0.4183880111235364*sqrt(abs(f))))*cos(1.160858784392945*sqrt(abs(f)))*cosh(1.160858784392945*sqrt(abs(f)))**2
    +(6426439.85085752*sqrt(abs(f))*cos(0.4183880111235364*sqrt(abs(f)))*sinh(0.4183880111235364*sqrt(abs(f)))
    -6426439.85085752*sqrt(abs(f))*cosh(0.4183880111235364*sqrt(abs(f)))*sin(0.4183880111235364*sqrt(abs(f))))
    *cosh(1.160858784392945*sqrt(abs(f)))*sin(1.160858784392945*sqrt(abs(f))))*(0.001905784463049503*abs(f)
    *(8.0*10**9*cos(0.4183880111235364*sqrt(abs(f)))*cosh(0.4183880111235364*sqrt(abs(f)))-8.0*10**9)
    *sin(0.009528922315247516*abs(f))+(267683.2445500172*abs(f)**1.5*cos(0.4183880111235364*sqrt(abs(f)))
    *sinh(0.4183880111235364*sqrt(abs(f)))+267683.2445500172*abs(f)**1.5*cosh(0.4183880111235364*sqrt(abs(f)))
    *sin(0.4183880111235364*sqrt(abs(f))))*cos(0.009528922315247516*abs(f))))/((cos(0.4183880111235364*sqrt(abs(f)))
    *cosh(0.4183880111235364*sqrt(abs(f)))-1)*((cos(0.4183880111235364*sqrt(abs(f)))
    *cosh(0.4183880111235364*sqrt(abs(f)))-1)*cos(1.160858784392945*sqrt(abs(f)))**2
    *cosh(1.160858784392945*sqrt(abs(f)))+(1-cos(0.4183880111235364*sqrt(abs(f)))*cosh(0.4183880111235364*sqrt(abs(f))))
    *cos(1.160858784392945*sqrt(abs(f))))*sinh(1.160858784392945*sqrt(abs(f)))+((1-cos(0.4183880111235364*sqrt(abs(f)))
    *cosh(0.4183880111235364*sqrt(abs(f))))*cos(1.160858784392945*sqrt(abs(f)))*cosh(1.160858784392945*sqrt(abs(f)))**2
    +(cos(0.4183880111235364*sqrt(abs(f)))*cosh(0.4183880111235364*sqrt(abs(f)))-1)*cosh(1.160858784392945*sqrt(abs(f)))
    *sin(1.160858784392945*sqrt(abs(f))))*cos(0.009528922315247516*abs(f)))
    -(1.72025027018307*10**12*f**2*sin(0.4183880111235364*sqrt(abs(f)))**2*sinh(0.4183880111235364*sqrt(abs(f)))**2)
    /(cos(0.4183880111235364*sqrt(abs(f)))*cosh(0.4183880111235364*sqrt(abs(f)))-1)**2;
    return g2

def g3(f):
    g3=(125149.2246406247*abs(f)**1.5*cos(1.160858784392945*sqrt(abs(f)))*sinh(1.160858784392945*sqrt(abs(f)))
    *sin(0.003890567728186338*abs(f))+125149.2246406247*abs(f)**1.5*cosh(1.160858784392945*sqrt(abs(f)))
    *sin(1.160858784392945*sqrt(abs(f)))*sin(0.003890567728186338*abs(f))-2.8786405512746 53*10**7*abs(f)
    *cos(1.160858784392945*sqrt(abs(f)))*cosh(1.160858784392945*sqrt(abs(f)))*cos(0.003890567728186338*abs(f))
    +2.8786405512746 53*10**7*abs(f)*cos(0.003890567728186338*abs(f)))/((cos(1.160858784392945*sqrt(abs(f)))
    *cosh(1.160858784392945*sqrt(abs(f)))-1)*sin(0.003890567728186338*abs(f)));
    return g3

def g4(f):
    g4=-((2321717.56878589*sqrt(abs(f))*cos(0.9123903877498877*sqrt(abs(f)))*cosh(0.9123903877498877*sqrt(abs(f)))
    *cos(1.160858784392945*sqrt(abs(f)))*sinh(1.160858784392945*sqrt(abs(f)))-2321717.56878589*sqrt(abs(f))
    *cos(1.160858784392945*sqrt(abs(f)))*sinh(1.160858784392945*sqrt(abs(f)))-2321717.56878589*sqrt(abs(f))
    *cos(0.9123903877498877*sqrt(abs(f)))*cosh(0.9123903877498877*sqrt(abs(f)))*cosh(1.160858784392945*sqrt(abs(f)))
    *sin(1.160858784392945*sqrt(abs(f)))+2321717.56878589*sqrt(abs(f))*cosh(1.160858784392945*sqrt(abs(f)))
    *sin(1.160858784392945*sqrt(abs(f)))+27594.24587341124*sqrt(abs(f))*cos(0.9123903877498877*sqrt(abs(f)))
    *sinh(0.9123903877498877*sqrt(abs(f)))*cos(1.160858784392945*sqrt(abs(f)))*cosh(1.160858784392945*sqrt(abs(f)))
    -27594.24587341124*sqrt(abs(f))*cosh(0.9123903877498877*sqrt(abs(f)))*sin(0.9123903877498877*sqrt(abs(f)))
    *cos(1.160858784392945*sqrt(abs(f)))*cosh(1.160858784392945*sqrt(abs(f)))-27594.24587341124*sqrt(abs(f))
    *cos(0.9123903877498877*sqrt(abs(f)))*sinh(0.9123903877498877*sqrt(abs(f)))+27594.24587341124*sqrt(abs(f))
    *cosh(0.9123903877498877*sqrt(abs(f)))*sin(0.9123903877498877*sqrt(abs(f))))*(5466.032505438206*abs(f)**1.5
          *cos(0.9123903877498877*sqrt(abs(f)))*sinh(0.9123903877498877*sqrt(abs(f)))*sin(0.009528922315247516*abs(f))
          +5466.032505438206*abs(f)**1.5*cosh(0.9123903877498877*sqrt(abs(f)))*sin(0.9123903877498877*sqrt(abs(f)))
```

```
        *sin(0.0095289223152475 16*abs(f))-1.524627570439603*10**7*abs(f)*cos(0.9123903877498877*sqrt(abs(f)))
        *cosh(0.9123903877498877*sqrt(abs(f)))*cos(0.0095289223152475 16*abs(f))+1.524627570439603*10**7*abs(f)
        *cos(0.0095289223152475 16*abs(f)))/((cos(0.9123903877498877*sqrt(abs(f)))
        *cosh(0.9123903877498877*sqrt(abs(f)))-1)**2*(cos(1.160858784392945*sqrt(abs(f)))
        *cosh(1.160858784392945*sqrt(abs(f)))-1)*sin(0.0095289223152475 16*abs(f)))
    -(1.508310449071199*10**8*f**2*sin(0.9123903877498877*sqrt(abs(f)))**2
      *sinh(0.9123903877498877*sqrt(abs(f)))**2)/(cos(0.9123903877498877*sqrt(abs(f)))
      *cosh(0.9123903877498877*sqrt(abs(f)))-1)**2;
    return g4




def trace(h,fmax,pas):
    N=int(fmax/pas)
    X=[(i+0.001)*pas for i in range(N) ]
    Y=[h((i+0.001)*pas) for i in range(N)]
    X1=[0.001*pas,fmax]
    Y1=[0,0]
    plt.plot(X,Y)
    plt.plot(X1,Y1,color='r')
    plt.xlabel('frequences')


def changement_signeg(h,fmax,fmin,pas):
    N=int((fmax-fmin)/pas)
    X=numpy.zeros(N)
    Y=numpy.zeros(N)
    Z=[]
    for i in range(N):
        X[i]=(i)*pas+fmin
        Y[i]=h(X[i])
        if (Y[i-1]/Y[i])<=0:
            Z.append(X[i])
    return Z



# x=scipy.optimize.root_scalar(g,method='newton',x0=1,x1=10)
```