DER Nikola Tesla, ARIA

Stage Centre borreli 3 mois

Analyse des caractéristiques des graphes pour améliorer les prédictions des GNN sur la consommation électrique.

Itai Zehavi
Encadré par
Argyris Kalogeratos et Eloi Campagne

Table des matières

1 Introduction et mise en contexte

Dans le cadre de mon stage, j'ai eu l'opportunité de travailler au **Centre Borelli**, laboratoire de recherche de l'ENS Paris-Saclay, sous la supervision d'Argyris Kalogeratos et d'Eloi Campagne. Mon stage s'inscrit dans le contexte de la *prédiction de la consommation électrique en France*, un sujet au cœur des enjeux énergétiques actuels. Ce domaine mobilise de nombreux algorithmes avancés capables d'anticiper la demande et d'optimiser la production d'électricité.

Parmi les algorithmes de prédiction les plus utilisés aujourd'hui, on trouve les *Random Forest*, les modèles *XGBoost* et les *Generalized Additive Models (GAM)*[1]. Les *GAM* se distinguent par leur capacité à fournir des prédictions performantes tout en restant interprétables, ce qui en fait l'état de l'art dans ce domaine. Cependant, l'algorithme réellement le plus performant repose sur une approche appelée *agrégation d'experts* [1]. Cette méthode combine les forces de plusieurs algorithmes en coordonnant leurs prédictions grâce à un modèle dédié, de manière à conserver uniquement les points forts de chaque approche.

Malgré ces avancées, des améliorations restent possibles, notamment en intégrant davantage d'informations dans les modèles. Aujourd'hui, les données utilisées pour la prédiction sont souvent moyennées à l'échelle nationale, ce qui masque des comportements locaux pourtant essentiels. Une meilleure prise en compte des spécificités régionales pourrait significativement améliorer les performances des algorithmes.

Une approche prometteuse pour relever ce défi est l'utilisation des *Graph Neural Networks (GNN)*[2]. Ces modèles permettent de représenter les interactions entre différentes régions sous forme de graphes et d'exploiter ces relations pour affiner les prédictions à l'échelle nationale tout en capturant les dynamiques locales. Les travaux récents montrent que les *GNN* peuvent produire des résultats très encourageants. Cependant, leur performance dépend fortement de la qualité du graphe initial modélisant les relations entre les régions. À ce jour, plusieurs graphes ont été construits en s'appuyant sur des méthodes spécifiques, décrites ci-dessous [2] :

- **Graphe 0 : Corrélation.** Ce graphe est construit en calculant les coefficients de corrélation entre les signaux temporels réduits (charge et température) pour chaque paire de régions. Cette méthode capture les relations linéaires fortes entre les séries temporelles.
- **Graphe 1 : Distance entre splines (Distspline).** Les séries temporelles sont d'abord modélisées à l'aide de splines (par exemple, des splines cubiques), qui permettent de lisser les variations locales. La matrice d'adjacence est ensuite obtenue en calculant les distances euclidiennes entre les splines associées à chaque région.
- **Graphe 2 : Distance entre splines normalisées (Distspline2).** Semblable à *Distspline*, mais en appliquant une normalisation préalable ou en utilisant une métrique différente (par exemple, la corrélation entre les splines).
- **Graphe 3 : DTW (Dynamic Time Warping).** Une matrice d'adjacence est construite en utilisant l'algorithme DTW, qui permet d'évaluer la similarité entre séries temporelles en tenant compte de décalages temporels non linéaires. Cette méthode est particulièrement adaptée pour des séries temporelles avec des variations de phase.
- **Graphe 4 : GL3SR (Graph Laplacian with Smooth and Bandlimited Regularization).** Ce graphe repose sur deux hypothèses :

— Les séries temporelles sont *lisses* sur le graphe : les nœuds proches dans le graphe devraient avoir des valeurs similaires.
— Les signaux sont limités en bande dans le domaine spectral, ce qui implique une représentation éparse.

Une matrice de Laplacien de graphe est régularisée pour respecter ces contraintes.

— **Graphe 5 : Précision.** Ce graphe est basé sur l'inverse de la matrice de covariance des séries temporelles (matrice de précision), qui représente les relations conditionnelles entre les régions.
— **Graphe 6 : Espace.** Ce graphe est construit en fonction des distances géographiques entre les régions, sans prendre en compte les séries temporelles. Les régions proches géographiquement sont connectées.

Les *GNN* ont montré des performances variées selon les graphes, comme illustré dans la figure ci-dessous :

Table 5: Numerical performance in MAPE (%) and RMSE (MW) at national level on the test set.

Model	Real Dataset		Synthetic Dataset ($\boldsymbol{\Sigma} = \boldsymbol{\rho}(\mathbf{W}_\lambda)$)		Synthetic Dataset ($\boldsymbol{\Sigma} = \boldsymbol{I}$)	
	MAPE (%)	RMSE (MW)	MAPE (%)	RMSE (MW)	MAPE (%)	RMSE (MW)
GAM-Regions	**1.48**	**1018**	**1.11**	**662**	**1.75**	**1043**
Feed Forward	1.54	1071	3.82	3141	4.49	3213
GCN-identity	5.66	3949	1.43	834	2.16	1259
GCN-space	2.07	1452	1.26	749	1.98	1169
GCN-distsplines	2.04	1404	1.29	764	2.01	1185
GCN-gl3sr	5.95	4210	**1.25**	**743**	**1.97**	**1160**
GCN-dtw	**1.82**	**1276**	1.26	753	1.99	1171
SAGE-identity	4.38	3021	1.25	755	**1.78**	**1066**
SAGE-space	1.96	1350	1.29	778	1.85	1112
SAGE-distsplines	2.06	1410	1.22	741	1.84	1116
SAGE-gl3sr	**1.78**	**1234**	**1.15**	**701**	1.92	1171
SAGE-dtw	1.90	1335	1.21	735	1.86	1127
Mixture (Baseline)	1.31	925	1.11	662	1.76	1044
Mixture (GNNs)	1.48	1092	1.12	677	1.98	1171
Mixture (Baseline + GNNs)	**1.13**	**844**	**1.08**	**647**	**1.76**	**1050**

FIGURE 1 – Performances des modèles GCN et GNN SAGE sur différents graphes. Les performances sont évaluées en termes d'erreur de prédiction RMSE (MW) et d'erreur relative MAPE (%).[2]

Nous observons ici que le graphe construit avec la méthode **GL3SR** [6] semble fournir les meilleurs résultats, tandis que le graphe sans aucune liaison entre les nœuds affiche une erreur deux fois plus élevée. Cela souligne l'importance cruciale de disposer d'un graphe bien construit pour optimiser les performances des *GNN*.

L'objectif de mon stage est donc de répondre à une question importante : *comment évaluer si un graphe est de bonne qualité pour la prédiction avec des GNN avant même de les entraîner ?* Actuellement, la seule méthode pour juger de la pertinence d'un graphe est d'entraîner un modèle complet et d'en évaluer les performances finales, ce qui est coûteux en temps et en énergie. L'enjeu est donc de trouver une manière plus efficace d'évaluer les graphes.

Dans un premier temps, nous allons présenter les données utilisées. Nous disposons d'un graphe représentant les régions françaises, composé de 12 nœuds reliés par des liens pondérés. Ces liens modélisent les interactions entre les régions. La structure générale de ces graphes est illustrée ci-dessous :

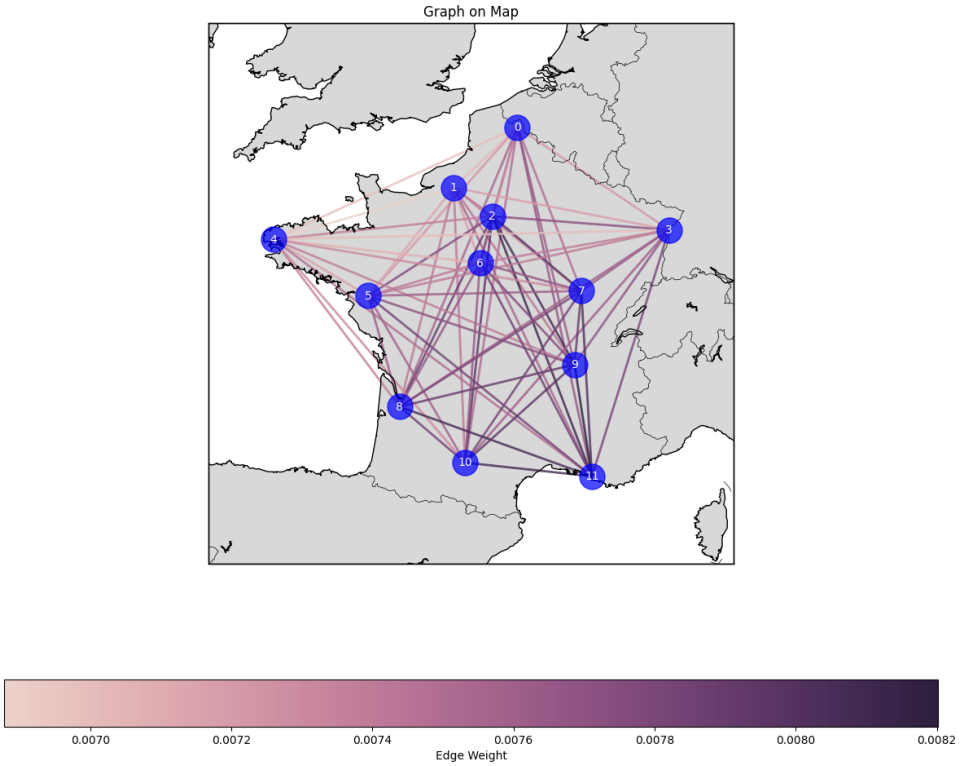


FIGURE 2 – Exemple de graphe représentant les régions françaises.[2]

Chaque nœud du graphe est enrichi par un ensemble de données, notamment des séries temporelles comme : la consommation électrique (*load*), la température moyenne (*temp*), la nébulosité (*nebu*), la vitesse du vent (*wind*), la température maximale journalière (*TempMax*), la température minimale journalière (*TempMin*), ainsi que bien d'autres variables. Ces six séries temporelles sont représentées ci-dessous :

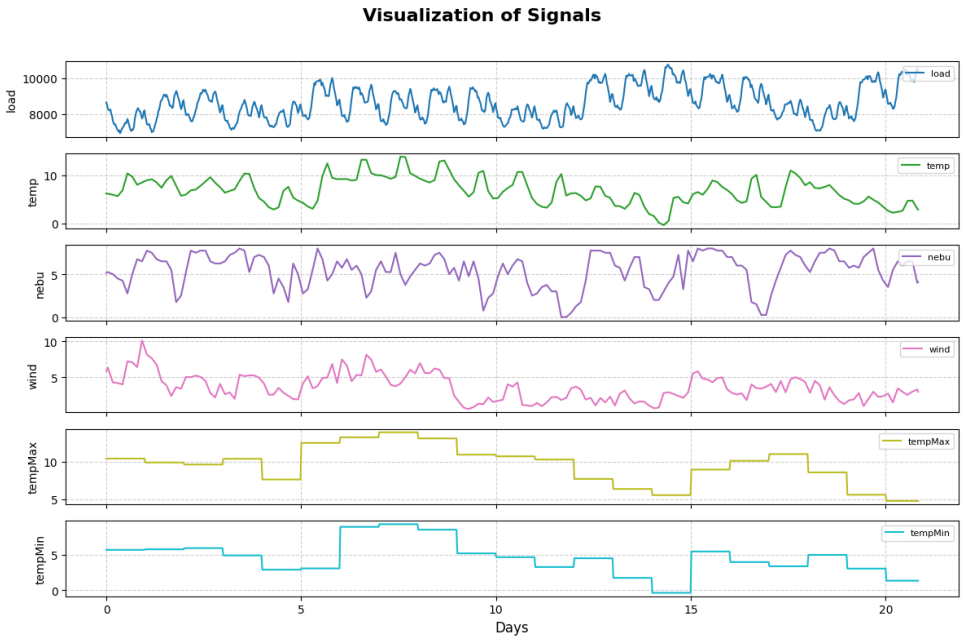


FIGURE 3 – Exemple de séries temporelles associées aux nœuds du graphe.

Dans ce stage, la démarche s'est organisée autour de plusieurs étapes méthodologiques pour répondre à la problématique : comment évaluer la qualité d'un graphe dans le cadre de la prédiction avec des *Graph Neural Networks (GNN)* ? Ces étapes sont présentées comme suit :

1. **Outils mathématiques pour les graphes à signal**
   Une première partie théorique a permis d'introduire les outils mathématiques nécessaires pour travailler avec les graphes.
2. **Application des *features* aux signaux bruts**
   Les *features* définies dans la bibliographie sont applicables uniquement sur des scalaires. Cette étape vise donc à appliquer directement ces différentes *features* sur les signaux, en les calculant pour chaque type de signal à chaque pas de temps.
3. **Application des *features* sur des données compressées**
   Ensuite, nous cherchons à généraliser ces *features* aux séries temporelles. Finalement, il suffit de généraliser les notions de distance et d'énergie. Cependant, il existe différentes manières de mesurer une distance entre des signaux, et nous utilisons donc diverses méthodes qui capturent des informations différentes pour calculer nos *features*.

À chaque étape, nous appliquerons les *features* aux graphes existants sur lesquels un *Graph Neural Network* (*GNN*) a déjà été entraîné. L'objectif sera d'identifier les caractéristiques des graphes qui influencent leur qualité pour la tâche de prédiction. Cela nous permettra de mieux comprendre quels aspects structurels ou dynamiques sont déterminants pour améliorer les performances des *GNN*.

2 Outils mathématiques pour les graphes à signal

Dans cette partie nous allons formaliser certains outils mathématiques des graphes à signal que nous utiliserons au cours de ce stage.

2.1 Laplacien combinatoire

Définition : Soit un graphe pondéré non orienté avec un Laplacien combinatoire L défini par [9] :

$$\mathcal{L} = D - W$$

où :
- D est la matrice des degrés, dont chaque élément diagonal $D_{i,i}$ est égal à la somme des poids des arêtes connectées au sommet i, soit $D_{i,i} = \sum_j W_{i,j}$.
- W est la matrice d'adjacence pondérée, où chaque élément $W_{i,j}$ représente le poids de l'arête entre les sommets i et j.

Propriété : l'action du Laplacien sur le signal f au sommet i est donnée par [9] :

$$(Lf)_i = \sum_{j \in \mathcal{N}_i} W_{i,j}(f(i) - f(j))$$

où $\mathcal{N}_i$ représente l'ensemble des sommets connectés au sommet i

La démonstration est faite dans l'annexe 6.1.

Interprétation : Cette expression montre que $(\mathcal{L}f)_i$ mesure la variation locale du signal f. Notamment la i^e ligne correspond à la variation entre le sommet i et ses voisins, pondérée par les poids des arêtes. Si f est constant sur tous les sommets voisins de i, alors $(\mathcal{L}f)_i = 0$.

2.2 Transformée de fourier sur Graphe (GFT)

En faisant l'analogie avec la transformée de Fourrier d'une fonction, nous pouvons définir celle sur un Graphe de signal.

Définition : Définition : Transformée de Fourier sur Graphe (GFT)
La Transformée de Fourier sur Graphe (GFT) d'un signal f sur le graphe est définie par [9] :

$$\hat{f}(\lambda_l) = \langle f, u_l \rangle = \sum_{i=1}^{N} f(i) u_l(i),$$

où u_l est le l-ème vecteur propre du Laplacien $\mathcal{L}$ et λ_l la l-ème valeur propre associée. De manière générale :

$$\hat{f} = U^T f$$

Propriété : Transformée de Fourier Inverse sur Graphe [9]
La transformée de Fourier inverse sur graphe permet de reconstruire le signal f en fonction des vecteurs propres du Laplacien :

$$f(i) = \sum_{l=0}^{N-1} \hat{f}(\lambda_l) u_l(i).$$

La démonstration est faite à l'annexe 6.2.

Interprétation :
Afin de bien comprendre ce que signifie une transformé de Fourrier sur graphe nous devons introduire le théorème de **Courant-fischer**. **théorème de Courant-Fischer** nous permet de les définir de manière itérative à l'aide du quotient de Rayleigh comme suit :

$$\lambda_0 = \min_{f \in \mathbb{R}^N, \|f\|_2 = 1} \left\{ f^\top \mathcal{L} f \right\},$$

et pour $l = 1, 2, \ldots, N-1$,

$$\lambda_l = \min_{\substack{f \in \mathbb{R}^N \\ \|f\|_2 = 1 \\ f \perp \mathrm{span}(u_0, \ldots, u_{l-1})}} \left\{ f^\top \mathcal{L} f \right\},$$

où le vecteur propre u_l est le minimiseur du l-ème problème.
Le théorème est très simple à démontrer : Annexe 6.3.
Ainsi la valeurs propre minimal correspond à la valeur minimal $f_0^T L f_0$. La deuxième valeurs propre correspond à la valeurs minimal de $f_1^T L f_1$ tel que $f_1^T f_0 = 0$ et ainsi de suite. En se basant sur cette vision Voici quelques interprétations clés :

- **Fréquences sur le graphe :** Les petites valeurs propres λ_l correspondent aux fréquences basses et sont associées aux variations lentes du signal sur le graphe. Si le signal est lisse (c'est-à-dire qu'il varie peu entre sommets connectés), la majorité de son énergie se concentre dans les coefficients $\hat{f}(\lambda_l)$ associés aux petites valeurs propres.
- **Modes de variation :** Chaque vecteur propre u_l représente un mode de variation du signal sur le graphe. Pour les fréquences basses (petites valeurs propres), u_l est généralement une fonction lisse sur le graphe. Pour les fréquences hautes (grandes valeurs propres), u_l présente des variations rapides entre nœuds connectés.
- **Applications :** En analysant les coefficients $\hat{f}(\lambda_l)$ dans le domaine spectral, il est possible d'extraire des informations sur la structure du signal, d'appliquer des filtres (passe-bas, passe-haut), et de réaliser des tâches comme le débruitage, la compression, et la détection d'anomalies sur le graphe.

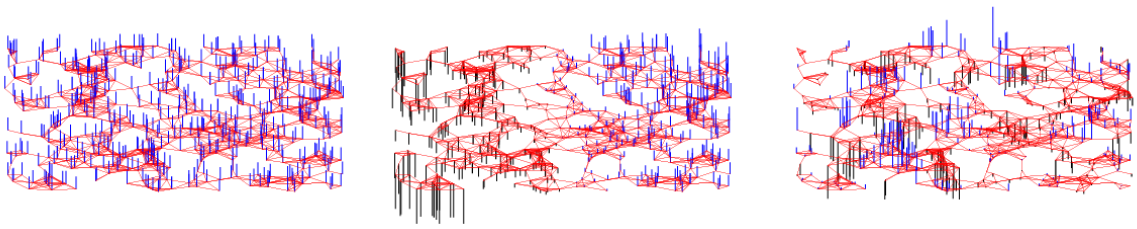


FIGURE 4 – Exemple de vecteur propre sur un graphe. U_0 est un vecteur propre associer à une variation quasi nulle. U_1 est un vecteur propre associer à une variation lente. $U_5 0$ est un vecteur propre associer à une variation rapide. [9]

2.3 Notion de dérivée et "smoothness" sur un graphe

Définition : Dérivée partielle sur un graphe [9]
La dérivée partielle d'un signal f au sommet i par rapport à une arête (i, j) connectant les sommets i et j est définie par :

$$\frac{\partial f}{\partial e_{i,j}} := W_{i,j} \left(f(j) - f(i) \right),$$

où $W_{i,j}$ est le poids de l'arête entre les sommets i et j. Cette dérivée mesure la variation du signal f le long de l'arête reliant i et j.

Définition : Gradient d'un signal sur un graphe [9]
Le gradient sur graphe du signal f au sommet i est le vecteur défini par :

$$\nabla_i f := \left(\frac{\partial f}{\partial e_{i,j}} \right)_{j \in \mathcal{N}_i},$$

où $\mathcal{N}_i$ représente l'ensemble des sommets voisins de i. Ce gradient regroupe les dérivées partielles de f par rapport à chaque arête connectée à i, capturant ainsi la variation locale du signal autour du sommet i.

Définition : Norme locale du gradient [9]
La norme locale du gradient au sommet i est définie par :

$$\|\nabla_i f\|_2 := \left(\sum_{j \in \mathcal{N}_i} \left| \frac{\partial f}{\partial e_{i,j}} \right|^2 \right)^{\frac{1}{2}} = \left(\sum_{j \in \mathcal{N}_i} W_{i,j}^2 \left(f(j) - f(i) \right)^2 \right)^{\frac{1}{2}}.$$

Cette norme fournit une mesure de la variation locale du signal au sommet i : elle est faible lorsque f a des valeurs similaires sur les sommets voisins de i, et élevée dans le cas contraire.

Les définitions précédentes permettent d'introduire la notion de smoothness avec la forme p-Dirichlet.

Définition : Forme discrète de Dirichlet pour la variation totale sur un graphe [9]
La forme discrète p-Dirichlet d'un signal f est définie par :

$$S_p(f) := \frac{1}{2} \sum_{i \in V} \sum_{j \in \mathcal{N}_i} (W_{i,j} \left| f(j) - f(i) \right|^2)^{\frac{P}{2}}.$$

Pour $p = 1$, $S_1(f)$ mesure la variation totale du signal sur le graphe :

$$S_1(f) = \frac{1}{2} \sum_{i \in V} \sum_{j \in \mathcal{N}_i} W_{i,j}^{\frac{1}{2}} \left| f(j) - f(i) \right|,$$

tandis que pour $p = 2$, on a :

$$S_2(f) = \frac{1}{2} \sum_{i \in V} \sum_{j \in \mathcal{N}_i} W_{i,j} \left(f(j) - f(i) \right)^2 = f^\top \mathcal{L} f,$$

où $S_2(f)$ est connue sous le nom de forme quadratique laplacienne du graphe.

Propriété : La Smoothness est la forme 2-Dirichlet définie précédemment :

$$S_2(f) = \frac{1}{2} \sum_{i \in V} \sum_{j \in \mathcal{N}_i} W_{i,j} \left(f(j) - f(i) \right)^2 = f^\top \mathcal{L} f,$$

Nous utiliserons cette *feature* pour caractériser nos graphes par la suite.

Interprétation : La *smoothness* sera l'une des caractéristiques que nous utiliserons par la suite. Celle-ci exprime les variations globales du graphe. En effet, une faible smoothness indique que le graphe contient des liens connectant des nœuds ayant des valeurs similaires, tandis qu'une forte smoothness signifie que le graphe relie des nœuds aux valeurs très différentes.

2.3.1 Laplacien Normalisé

Le Laplacien Normalisé est introduit de manière à "équilibré un graphe" dans le sens où si un nœuds contient beaucoup de lien avec des points élevé alors toute l'énergie du graphes sera très dépendantes de ce nœuds. Ainsi la bibliographie recommande de normaliser chaque poids $W_{i,j}$ par les degrés du nœuds i et j. Ce laplacien permet de mieux étudier l'ensemble de la structure du graphe. Surtout, cette normalisation permet de pouvoir comparer plusieurs graphes car nous assurons que tous les graphes ont la même "Energie" totale.

Définition : Laplacien Normalisé [9]
Soit un graphe pondéré non orienté avec un Laplacien combinatoire L et une matrice des degrés D. Le Laplacien normalisé, noté $\mathcal{L}$, est défini par :

$$\mathcal{L} := D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$$

où $L = D - W$ et W est la matrice d'adjacence pondérée du graphe.

Propriété :
- Les valeurs propres de $\mathcal{L}$ pour un graphe connexe satisfont $0 = \lambda_0 < \lambda_1 \leq \cdots \leq \lambda_{\max} \leq 2$.
- $\lambda_{\max} = 2$ si et seulement si le graphe est biparti, c'est-à-dire que l'ensemble des sommets peut être partitionné en deux sous-ensembles tels que chaque arête connecte un sommet d'un sous-ensemble à un sommet de l'autre sous-ensemble.
- Les vecteurs propres associés aux valeurs propres élevées de $\mathcal{L}$ tendent à présenter davantage de changements de signe, ce qui correspond à des "fréquences" plus élevées dans le domaine spectral du graphe.
- Contrairement au Laplacien combinatoire, le vecteur propre associé à la valeur propre zéro pour le Laplacien normalisé n'est pas nécessairement constant.

Interprétation : Laplacien Normalisé
La matrice de Laplacien normalisé $\mathcal{L}$ capture la structure locale et globale du graphe tout en prenant en compte la variation des degrés entre nœuds. En normalisant les poids, $\mathcal{L}$ permet une comparaison plus équitable des variations de signal entre graphes de tailles ou de densités différentes, facilitant l'analyse spectrale.

Cependant, dans notre cas, cette normalisation n'est pas adaptée. En effet, elle altère le sens physique de nos graphes. Certaines régions de France sont plus "indépendantes" que d'autres et, par conséquent, sont moins connectées dans le graphe par rapport à des régions plus centrales. Il est donc essentiel de conserver cette proportionnalité dans les degrés de nos nœuds. Toutefois, la normalisation reste importante, car elle nous permettra de comparer nos graphes de manière cohérente.

Définition : Nous définissons donc une nouvelle méthode pour normaliser nos graphes :

$$L_{\text{normalisé}} = \frac{\mathcal{L}}{\sum_{i=1}^{N} \sum_{j=1}^{N} w_{i,j}}$$

où L est le laplacien et $w_{i,j}$ représente les poids de la matrice d'adjacence.

2.4 Local smoothness

La local smoothness permet d'identifier la smoothness d'un nœuds. Ce qui permet de comprendre les différentes structure locales d'un graphe [4].

Définition : La **local smoothness** [4] d'un signal $\mathbf{x}$, défini sur les nœuds d'un graphe $G = (V, E)$, mesure les variations locales du signal autour d'un nœud n. Elle peut être exprimée de deux manières :

— **Formulation terme à terme :**

$$\lambda(n) = \sum_{m \in \mathcal{N}(n)} w_{nm} \big(x(n) - x(m)\big)^2,$$

où :
— $\mathcal{N}(n)$ est l'ensemble des voisins du nœud n,
— w_{nm} est le poids de l'arête connectant n et m,
— $x(n)$ et $x(m)$ sont les valeurs du signal aux nœuds n et m.

— **Formulation matricielle :**

$$\lambda(n) = \mathbf{x}^\top L_i \mathbf{x},$$

où :
— $\mathcal{L}_i$ est le laplacien local pour le nœud n, défini comme :

$$(\mathcal{L}_i)_{k,l} = \begin{cases} w_{nm}, & \text{si } k = l = n, \\ -w_{nm}, & \text{si } k = n \text{ et } l = m, \\ 0, & \text{sinon.} \end{cases}$$

— $\mathbf{x}$ est le vecteur des valeurs du signal sur tous les nœuds.

La local smoothness $\lambda(n)$ reflète ainsi les variations locales du signal au nœud n, en tenant compte des connexions pondérées par les arêtes du graphe.

Interprétation : En simple la local smoothness n'est rien d'autre que la smoothness d'un noeud n avec ses voisins.

Interprétation : **Dans le cas d'un signal monocompsante** (un signal proportionnel à un vecteur propre de L) alors la locale smoothness est simplement la fréquence du signal.
La notion fréquentielle et de variation locale sur un graphe est très lien. Dans le sens ou une haute fréquence signifie beaucoup de variation localement et inversement. De plus certain noeuds peuvent concentrer plus d'Energie à une fréquence que d'autre noeuds. Avec la notion de locale smoothness nous pouvons donc étudier les fréquences majoritaire localement.

Interprétation : **Dans le cas d'un signal multi-composantes :** La local smoothness donne la moyenne pondérée des fréquences du nœuds.

2.5 Bandwith (la largeur de bande spectral)

Afin d'expliquer cette notion nous allons devoir introduire la notion de distribution vertex-frequency. Ces distributions vont permettre de faire plusieurs chose dont :

- Estimer la largueur de bande du signal : à quelle point la dispersion d'un signal sur graphe dans le domaine spectral (du graphe).
- Estimer la fréquence majoritaire d'un nœuds. (très utile dans le cas d'un signal multi-composante, donc le cas général).

2.5.1 Distributions vertex-frequency

Il existe plusieurs distributions différentes vertex-frequency ayant chacune des propriétés différentes. Par exemple, certaine distribution limite les interférence, d'autres sont simple à calculer...

Définition : **définition d'une distribution vertex-fréquence** [4] Une distribution vertex-fréquence $G(n,k)$ est cohérente avec la définition de la local smoothness si elle satisfait :

$$\frac{\sum_{k=1}^{N} \lambda_k G(n,k)}{\sum_{k=1}^{N} G(n,k)} = \lambda(n).$$

Cela signifie que $G(n,k)$ doit représenter correctement la contribution de chaque valeur propre λ_k au nœud n. Ainsi, la local smoothness peut être récupérée comme une moyenne pondérée des λ_k, où les poids sont définis par $G(n,k)$.

Interprétation : Respecter la local smoothness pour $G(n,k)$ signifie que la distribution encode correctement l'énergie locale du signal dans le domaine spectral. Cela revient à dire que $G(n,k)$ agit comme un "filtre" qui pondère les contributions spectrales λ_k en fonction de leur importance pour le signal sur le nœud n. Physiquement :

- λ_k correspond à l'intensité de la variation spectrale associée au mode k sur le graphe.
- $G(n,k)$ mesure comment chaque mode spectral k contribue à la structure locale autour du nœud n.
- La smoothness locale $\lambda(n)$ reflète une moyenne pondérée des λ_k, reliant directement les variations locales du signal à la topologie du graphe.

2.5.2 Bandwith

Définition : Bandwith (bande passante) :
La bande passante de la local smoothness pour une distribution vertex-fréquence $G(n,k)$ est définie comme :

$$\text{Bande passante} = \sqrt{\frac{\sum_{k=1}^{N}(\lambda_k - \lambda(n))^2 G(n,k)}{\sum_{k=1}^{N} G(n,k)}}.$$

Cette bande passante mesure la dispersion de $G(n,k)$ autour de $\lambda(n)$, indiquant si l'énergie est concentrée ou étalée sur plusieurs fréquences.

Interprétation : Implications physiques.
- Une **faible bande passante** Indique que $G(n,k)$ (peut être l'énergie du signal à un nœud et une fréquence données) est concentré autour de très peu de fréquences au nœud n.
- Une **large bande passante** traduit un signal complexe, $G(n,k)$ est dispersé sur beaucoup de fréquence au noeud n.
- Nous utiliserons la bandwidth par la suite comme *feature* pour caractérisé nos graphes.

Nous faisons l'exemple du calcul d'une bande passante dans l'annexe : 6.6.

3 Application des *features* aux signaux bruts

Avant de commencer cette partie, rappelons que l'objectif principal de ce stage est d'identifier les caractéristiques des graphes ayant une influence significative sur les performances des GNN appliqués à ces derniers.

Dans cette optique, nous avons défini, dans la partie précédente, les outils mathématiques des graphes à signal nécessaires pour notre analyse. Ces outils permettent de calculer des *features* sur des données scalaires associées aux nœuds. Cependant, ces *features* doivent être appliquées à chaque série temporelle et pour chaque instant donné. Cela entraîne une première difficulté majeure liée à la gestion d'un volume massif d'informations : avec 10 graphes, 30 séries temporelles et 80 000 instants, ce sont plusieurs dizaines de millions de *features* qui doivent être calculées, analysées et visualisées.

Pour rendre ce processus exploitable, nous nous fixons trois objectifs principaux :

1. Identifier les *features* les plus pertinentes afin de concentrer l'analyse sur les informations essentielles.
2. Développer une méthode efficace et claire pour visualiser les *features*, tout en préservant la lisibilité.
3. Sélectionner une série temporelle représentative pour illustrer le calcul et l'interprétation des *features*.

3.1 Identification des *features* pertinentes

Dans le cadre de l'apprentissage d'un réseau de neurones de graphes (GNN), un graphe de qualité est celui qui reflète une structure cohérente, capable d'améliorer les performances d'un modèle. Pour évaluer cette qualité, deux *features* principales ont été retenues : la *smoothness* et la *bandwidth*.

— **La *smoothness* (lissitude)** : Cette mesure quantifie la régularité du signal sur le graphe. Une faible *smoothness* indique une structure cohérente, où les nœuds connectés partagent des valeurs similaires. À l'inverse, une *smoothness* élevée traduit des connexions aléatoires ou non informatives, où les nœuds connectés présentent des variations importantes. [6]

— **La *bandwidth* (bande passante)** : Cette *feature* décrit la dispersion fréquentielle du signal dans le domaine spectral du graphe. Un signal ayant une faible bande passante concentre son énergie sur un nombre restreint de fréquences, ce qui est typique d'un graphe structuré. En revanche, une *bandwidth* élevée est souvent associée à des signaux bruités ou des graphes non structurés. [6]

Ces deux *features* sont calculées à deux échelles complémentaires :

— **Globale :** Une mesure réalisée sur l'ensemble du graphe, permettant d'évaluer sa structure générale.

— **Locale :** Une mesure réalisée sur chaque nœud et ses voisins immédiats, offrant une perspective plus fine et détaillée.

Bien que les *features* locales soient intéressantes pour analyser les variations à petite échelle, notre étude se concentre principalement sur l'analyse globale pour évaluer la cohérence générale des graphes.

3.2 Visualisation des *features*

L'affichage des *features* constitue une tâche complexe en raison du grand nombre de graphes, séries temporelles et instants temporels à analyser. Afin de rendre ces données compréhensibles et exploitables, nous adoptons une stratégie en deux étapes :

1. **Visualisation temporelle :** Nous affichons l'évolution des *features* (telles que la *smoothness* et la *bandwidth*) sur une période donnée, comme 7 jours. Pour mettre en évidence la dispersion des valeurs locales, nous utilisons les écarts-types locaux des *features*, représentés sous forme d'une enveloppe autour des courbes. Cette visualisation permet d'observer les variations dynamiques des *features* tout en intégrant une mesure de leur variabilité spatiale à chaque instant.

2. **Résumé statistique global :** Nous présentons les moyennes et écarts-types des *features* calculées sur l'ensemble des instants temporels pour chaque graphe. Ces informations sont visualisées sous forme de nuages de points, offrant une vue synthétique et comparative des différentes configurations de graphes et séries temporelles. Cette représentation facilite l'identification de tendances globales et de comportements atypiques.

3. **Comparaison avec des graphes aléatoires :** Pour contextualiser et interpréter les valeurs obtenues sur nos graphes, nous comparons celles-ci avec les densités de probabilité estimées à partir de graphes générés aléatoirement. Ces distributions sont obtenues à l'aide de la méthode de Monte-Carlo, qui consiste à générer un

grand nombre de graphes aléatoires respectant certaines propriétés structurelles, comme la *sparsity*. Cette étape permet de déterminer si les valeurs de *features* calculées sur nos graphes réels sont significativement différentes de celles attendues pour des graphes aléatoires, révélant ainsi leur caractère informatif ou structurellement cohérent.

Cette approche en trois volets combine une analyse des dynamiques temporelles, une synthèse statistique globale et une comparaison avec des graphes aléatoires. Elle garantit une compréhension approfondie des *features* tout en permettant une gestion efficace du volume de données. Ces méthodes offrent ainsi une représentation claire, informative et pertinente pour évaluer la qualité des graphes.

3.3 Identification des séries temporelles pertinentes

Chaque nœud des graphes contient plusieurs séries temporelles associées, représentant différents types de données telles que la température, la consommation électrique ou la vitesse du vent. Toutefois, toutes ces séries n'ont pas le même intérêt pour l'analyse.

Certaines séries, comme les dates, produisent des signaux constants, identiques pour tous les nœuds, et n'apportent donc aucune information pertinente. À l'inverse, la série temporelle `load` (consommation électrique) est particulièrement intéressante. Elle révèle des comportements similaires entre certaines régions, mettant en évidence des structures cohérentes dans les graphes. De même, les données météorologiques (température et vent) peuvent refléter des conditions climatiques partagées entre régions, influençant les comportements observés.

Cependant, pour simplifier l'analyse et illustrer les calculs, nous avons décidé de nous concentrer sur la série temporelle `load`. Elle sert ainsi de référence pour le calcul et la visualisation des *features*, offrant un cas représentatif pour explorer les caractéristiques des graphes étudiés. De plus les graphes évaluer durant ce stage ont été construit en partie grâce au séries temporelles de consommation électrique et de température [2].

3.4 Algorithme de calcul

Dans les sous parties parties précédentes nous avons déterminer les features et séries temporelles que nous allons utiliser. A présent, il est important de définir l'algorithme de calcul de ces features et surtout de définir des vérifications attestant de son bon fonctionnement. Nous avons donc réaliser un algorigramme.

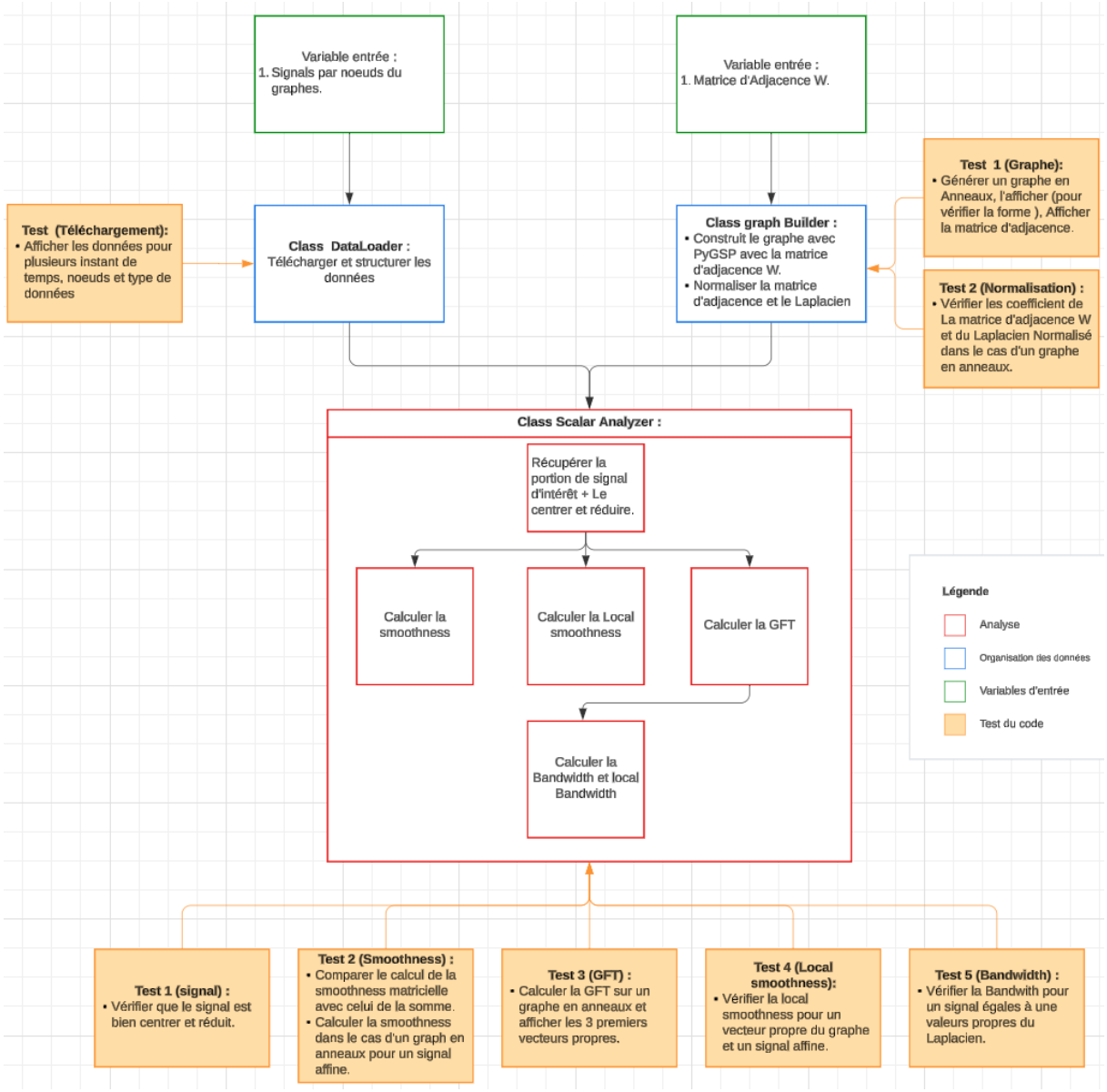


FIGURE 5 – Algorigramme de l'analyse des features appliqués au scalair.

1. **Variables d'entrée :**
   — Signal par nœuds du graphe.
   — Matrice d'adjacence W.
2. **Chargement et structuration des données :**
   — Utilisation de la classe `DataLoader`.
   — Téléchargement et structuration des données.
3. **Construction du graphe :**
   — Utilisation de la classe `Graph Builder`.
   — Construction du graphe avec PyGSP [5].
   — Normalisation de la matrice d'adjacence et du Laplacien.

Nous rappelons la normalisation du Laplacien définie dans la partie précé-

dente :

$$L_{\text{normalisé}} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$$

Cependant, cette normalisation n'est pas adaptée à notre contexte. Elle normalise les liens de chaque nœud indépendamment, c'est-à-dire que la somme des poids des liens d'un nœud avec ses voisins est égale à 1. Cela déforme la structure initiale du graphe, car certaines régions peuvent être naturellement plus indépendantes que d'autres. Il est donc crucial que la somme des poids des nœuds ne soit pas contrainte à une valeur unique, comme expliqué dans la partie précédente.

Pour conserver la structure globale du graphe, nous proposons une normalisation à l'échelle du graphe entier :

$$L_{\text{normalisé}} = \frac{L}{\sum_{i=1}^{N} \sum_{j=1}^{N} w_{i,j}}$$

où L désigne le Laplacien du graphe et W la matrice d'adjacence associée.

4. **Analyse scalaire :**
   — Utilisation de la classe `Scalar Analyzer`.
   — Centrage et réduction du signal f.

$$f_{normalis} = \frac{f - \mathbb{E}[f]}{\sigma_f}$$

   — Calcul de la **Smoothness**. Nous rappelons son expression :

$$S(f) = \frac{1}{2} \sum_{i \in V} \sum_{j \in \mathcal{N}_i} W_{i,j} \left(f(j) - f(i)\right)^2 = f^\top L f,$$

   Avec S ma smoothness et f le signal sur graphe.
   — Calcul de la **Graph Fourier Transform** (GFT). Nous rappelons l'expression :

$$\hat{f} = U^T f$$

   où $\hat{f}$ est la transformée de Fourier sur graphe (GFT) du signal f et U la matrice formée par la concaténation des vecteurs propres du Laplacien L.

   La matrice U est une matrice de changement de base orthogonale permettant de diagonaliser L. Les valeurs propres de L correspondent aux fréquences discrètes du graphe.
   — Calcul de la *Local Smoothness*. Nous rappelons l'expression de la local smoothness au noeud n.

$$\lambda(n) = \frac{\sum_{m \in \mathcal{N}(n)} w_{nm} (f(n) - f(m))^2}{f(n)^2}$$

   — Calcul de la **Bandwidth**. Nous rappelons l'expression de la Bandwidth.

$$\text{Bandwidth} = \sqrt{\frac{\sum_{k=1}^{N} (\lambda_k - \lambda(n))^2 |f(n) U_k(n)|^2}{\sum_{k=1}^{N} |f(n) U_k(n)|^2}}.$$

   avec $\lambda(n) = \frac{\sum_{k=1}^{n} \lambda_k * |f(n) U_k(n)|^2}{\sum_{k=1}^{n} |f(n) U_k(n)|^2}$ Cela permet de mesure la dispersion en fréquence du signal f au nœud n.

Ensuite, nous réalisons un ensemble de vérification que vous trouverez en annexe 6.7 afin d'attester du bon fonctionnement du code.

3.5 Affichage des résultats

Pour améliorer l'interprétabilité des mesures, il est nécessaire de définir des valeurs de référence. Ces valeurs peuvent être obtenues de différentes manières, comme en calculant les *features* sur des cas classiques ou bien sur des données totalement aléatoires. Dans notre contexte, l'absence de cas usuels pertinents nous amène à privilégier une approche basée sur des graphes aléatoires.

La méthodologie suivie est la suivante :

— Calcul des *features* sur un bruit blanc appliqué à des graphes générés aléatoirement, tout en s'assurant que ces graphes possède la même *sparsity*.

— Calcul des *features* sur un signal réel, tel que la série temporelle `load`, appliqué à des graphes aléatoires. À l'aide de la méthode de Monte-Carlo, nous estimons ensuite une distribution des *features* obtenues pour ces graphes aléatoires. Cette distribution sert de référence pour évaluer les graphes réels.

Si les valeurs de *features* des graphes réels s'écartent de manière significative de cette distribution, cela indique que les connexions présentes dans le graphe sont probablement cohérentes avec la série temporelle analysée. En revanche :

— Une *smoothness* anormalement basse pour `load` indique que le graphe connecte préférentiellement des régions présentant des séries temporelles similaires.

— Une *smoothness* anormalement élevée suggère que le graphe connecte volontairement des régions ayant des comportements très différents.

— Une valeur proche de la distribution aléatoire indique que les liens du graphe réel ont été établis sans structure informative particulière pour la série temporelle donnée.

Cette démarche permet de mettre en évidence les zones où les graphes réels présentent des informations significatives et d'identifier les liens les plus pertinents pour améliorer la qualité des graphes.

Pour faciliter la lisibilité et la correspondance entre les graphes, nous avons attribué des indices aux graphes générés conformément aux différentes méthodes mentionnées dans l'introduction [2] :

— Graphe 0 : Corrélation
— Graphe 1 : Distspline
— Graphe 2 : Distpline2
— Graphe 3 : DTW
— Graphe 4 : GL3SR
— Graphe 5 : Précision
— Graphe 6 : Spatial

3.5.1 Valeurs de référence

Dans un premier temps, nous présentons les résultats de la *smoothness* et de la *bandwidth* pour des graphes générés aléatoirement, le signal étant un bruit blanc gaussien centré et réduit. Théoriquement, nous nous attendons à obtenir une *smoothness* proche de $1 + \frac{1}{12}$, conformément à la démonstration mathématique présentée en annexe (6.8).

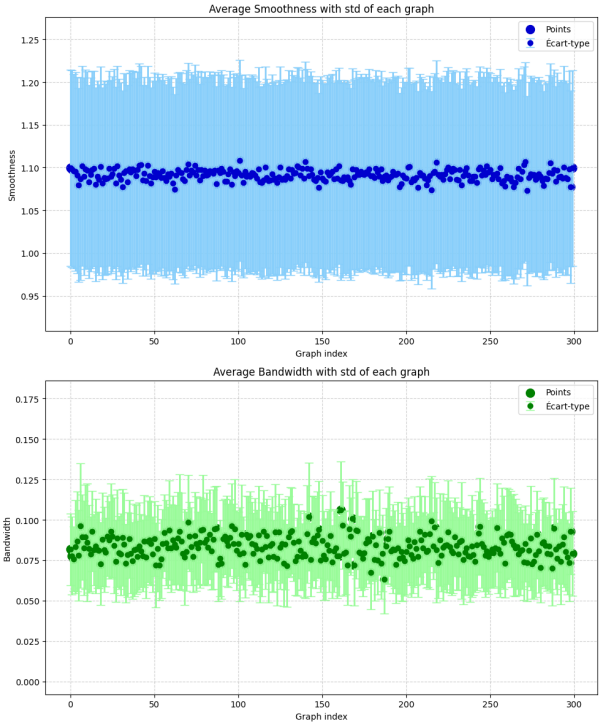


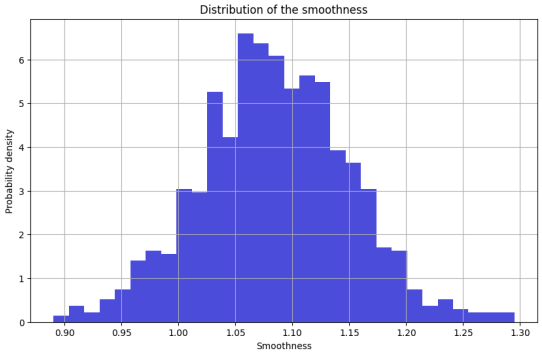
FIGURE 6 – Valeurs moyennes de la *smoothness* et de la *bandwidth* (calculées sur 500 instants) pour un bruit blanc appliqué à 100 graphes générés aléatoirement.

Sur la figure 6, nous observons que la *smoothness* converge vers une valeur moyenne de 1.08, tandis que la *bandwidth* tourne autour de 0.08. Ces deux valeurs servent de référence pour nos analyses ultérieures.

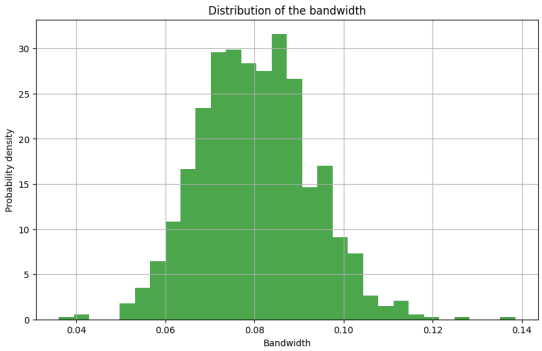
Une *smoothness* inférieure à 1.08 indique que le graphe est relativement lisse par rapport aux données, plus cette valeur se rapproche de 0, plus le graphe est cohérent avec les signaux. À l'inverse, une *smoothness* supérieure à 1.1 suggère des variations importantes entre les nœuds connectés, ce qui peut indiquer une structure aléatoire ou bruitée.

Un raisonnement similaire s'applique à la *bandwidth* : une faible valeur indique un signal concentré autour d'une fréquence, correspondant à un graphe structuré et cohérent. En revanche, une *bandwidth* élevée traduit un signal dispersé, souvent associé à un graphe bruité ou aléatoire.

Pour approfondir l'analyse, nous estimons ensuite la distribution des *features smoothness* et *bandwidth* pour des graphes aléatoires appliqués à la série temporelle `load`, en utilisant la méthode de Monte-Carlo. Cette étape consiste à générer 1000 graphes aléatoires, puis à calculer les distributions de ces *features* pour évaluer leurs comportements statistiques. Cela permet non seulement d'établir des valeurs de référence, mais également d'identifier d'éventuelles structures particulières dans les graphes réels.



(a) Distribution de la *smoothness* sur la série temporelle `load`.



(b) Distribution de la *bandwidth* sur la série temporelle `load`.

FIGURE 7 – Distributions estimées de la *smoothness* et de la *bandwidth* pour des graphes aléatoires appliqués à la série temporelle `load`.

Les distributions obtenues, représentées dans la figure 7, semblent suivre une loi normale, avec une moyenne d'environ 1.08 pour la *smoothness* et 0.08 pour la *bandwidth*. Ces résultats confirment que les graphes aléatoires présentent des valeurs de *features* centrées autour de ces moyennes, ce qui en fait des points de comparaison pertinents pour les graphes réels. Ces distributions de référence permettront de détecter les graphes dont les valeurs de *features* s'écartent significativement, indiquant ainsi des structures non aléatoires.

Enfin, pour explorer les relations entre les *features*, nous étudions la corrélation entre la *smoothness* et la *bandwidth*.

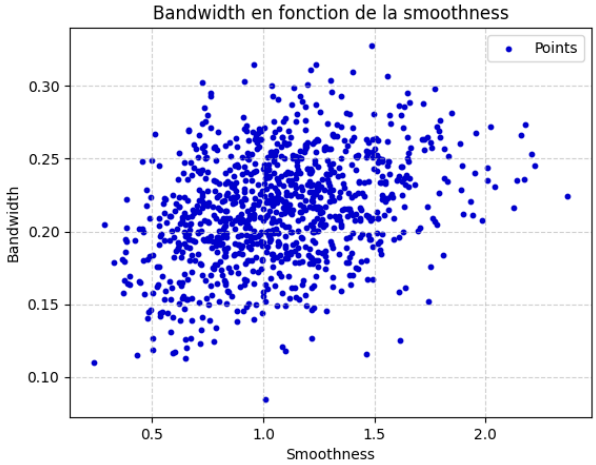


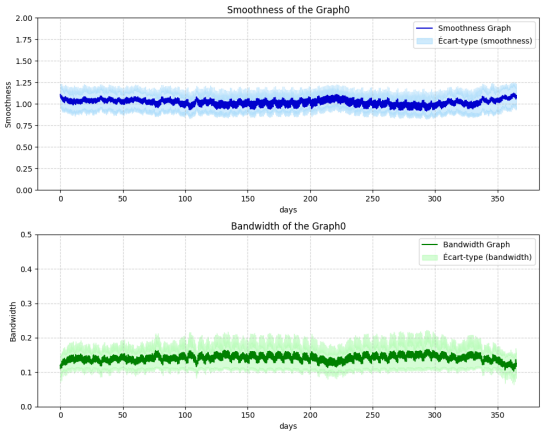
FIGURE 8 – Relation entre la *smoothness* et la *bandwidth* pour 1000 graphes générés aléatoirement.

Comme le montre la figure 8, la *smoothness* et la *bandwidth* semblent très peu corrélées. Cela suggère que ces deux *features* capturent des informations complémentaires et non redondantes sur les graphes. Cette indépendance renforce l'intérêt d'utiliser ces deux *features* conjointement pour une analyse plus exhaustive des graphes.

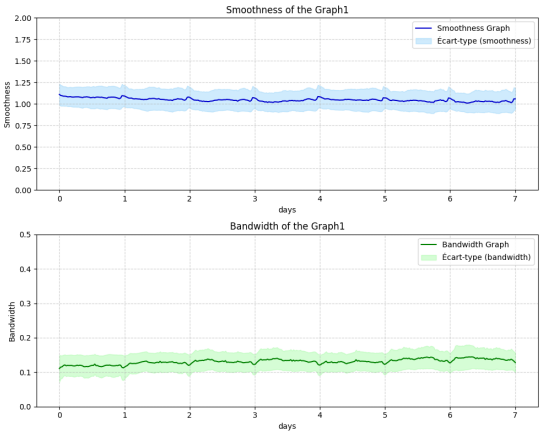
3.5.2 Résultats sur les graphes et données réels

Dans cette sous-section, nous analysons les *features* calculées sur des graphes réels afin de comprendre leur comportement et leur pertinence. L'objectif est de mettre en évidence les structures particulières des graphes à partir des séries temporelles réelles, en étudiant la stationnarité des *features*, leur distribution, et leur corrélation avec des propriétés non aléatoires.

Dans un premier temps, nous examinons la stationnarité des *features smoothness* et *bandwidth* au cours d'une année complète pour la série temporelle `load` sur le graphe 1.



(a) Évolution de la *smoothness* et de la *bandwidth* sur une année complète.



(b) Évolution de la *smoothness* et de la *bandwidth* sur une semaine complète.

FIGURE 9 – Comparaison des évolutions de la *smoothness* et de la *bandwidth* sur la série temporelle `load`.

Comme illustré dans la figure 9, la *smoothness* et la *bandwidth* montrent des variations non stationnaires, particulièrement marquées pendant la période estivale, où un pic est observé. Ce comportement suggère que le graphe 1 a été construit à partir de données qui reflètent principalement des conditions hors été. Cela met en évidence une sensibilité des *features* aux variations saisonnières, ce qui peut être crucial pour évaluer leur pertinence dans différents contextes temporels.

Ensuite, nous comparons les valeurs moyennes des *features* calculées sur une semaine, en les superposant à la densité de probabilité estimée à partir de graphes aléatoires. Cette étape vise à évaluer si les graphes réels se distinguent des graphes aléatoires en termes de structure.

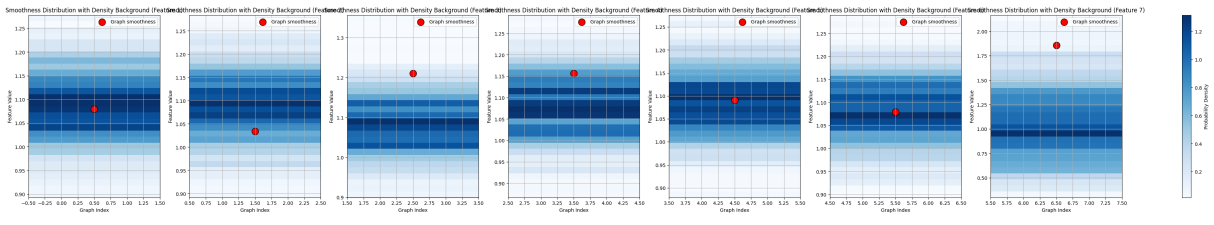


FIGURE 10 – Distribution de la *smoothness* sur la série temporelle `load`, superposée à la densité de probabilité estimée.

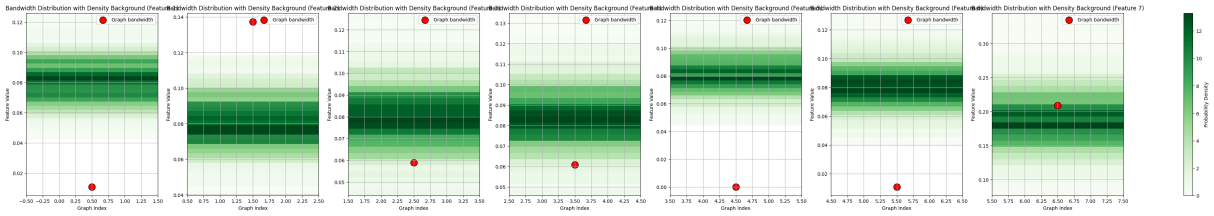


FIGURE 11 – Distribution de la *bandwidth* sur la série temporelle `load`, superposée à la densité de probabilité estimée.

Les figures 10 et 11 montrent que certains graphes présentent des valeurs de *features* significativement éloignées de celles attendues pour des graphes aléatoires. Par exemple, les graphes 2 et 6 affichent des valeurs atypiques de *smoothness*, indiquant une structure non aléatoire et potentiellement cohérente vis-à-vis de la série temporelle `load`. De manière similaire, pour la *bandwidth*, les graphes 0, 2, 4, et 5 révèlent également des connexions structurées qui diffèrent de celles des graphes générés aléatoirement.

Ces observations soulignent que certaines configurations de graphes sont capables de capturer des relations spécifiques dans les données, ce qui peut être un indicateur de leur pertinence pour les tâches d'apprentissage.

Enfin, une analyse plus détaillée consiste à identifier les séries temporelles pour lesquelles les graphes révèlent une structure non aléatoire. En corrélant ces observations avec les performances des GNN sur ces graphes, nous pouvons évaluer quelles *features* sont les plus importantes pour caractériser la qualité d'un graphe.

Conclusion partielle

L'analyse des *features* *smoothness* et *bandwidth* sur des graphes réels révèle leur sensibilité aux variations temporelles et leur capacité à capturer certaines structures non aléatoires dans les séries temporelles. Cependant, ces *features* sont limitées dans leur capacité à représenter l'ensemble des informations contenues dans les séries temporelles. En effet, elles se concentrent sur des propriétés locales ou globales à des instants précis, sans tenir compte de la dynamique complète des séries temporelles.

Cette limitation indique que nos *features* actuelles ne capturent probablement pas suffisamment d'informations pour caractériser pleinement les graphes et leurs interactions avec les données. Pour surmonter cette contrainte, il est nécessaire d'exploiter des méthodes capables de considérer les séries temporelles dans leur entièreté. Cela nous permettra d'analyser des signaux compressés qui intègrent les variations temporelles globales, tout en conservant des informations essentielles.

Ainsi, dans la partie suivante, nous allons introduire une approche basée sur des signaux compressés, visant à enrichir la description des graphes et à dépasser les limitations des *features* actuelles.

4 Applications des *features* sur les données compressées

Jusqu'à présent, les *features* que nous avons définies étaient calculables lorsque chaque nœud était porteur d'un unique scalaire. Cependant, dans le cadre de notre étude, chaque nœud contient plusieurs séries temporelles. Cette extension rend les notions classiques de *smoothness* et de *local smoothness* moins pertinentes.

En effet, si les séries temporelles associées aux nœuds sont très similaires mais présentent des décalages (déphasage), des bruits ou encore des contractions/dilatations temporelles, la valeur de la *smoothness* à un instant t pourrait être élevée, et ce, même si les séries temporelles globalement se ressemblent. Cela résulte du fait que la *smoothness* classique évalue des différences instantanées sans capturer les dynamiques globales des séries.

Ainsi, il devient nécessaire de trouver des méthodes pour généraliser nos résultats précédents dans ce cas particulier. Une première approche consiste à ajuster un modèle mathématique aux séries temporelles de chaque nœud, puis à calculer la *smoothness* sur les paramètres de ces modèles, et non directement sur les valeurs des séries temporelles.

L'objectif de cette partie est de compresser directement les données afin de réduire drastiquement le nombre de *features* à analyser, tout en restant cohérent vis-à-vis des signaux. En effet, pour généraliser les *features* définies dans la partie 2, il est nécessaire de définir une notion de distance entre signaux. Cependant, la distance euclidienne normalisée présente de nombreux défauts : elle est notamment très sensible aux déphasages, contractions, dilatations et au bruit. Il est donc crucial d'explorer différentes manières de calculer les distances entre signaux pour répondre à ces problématiques.

4.1 Pré-traitement

Dans un premier temps, nous centrons et réduisons tous les signaux afin de rendre les mesures des *features* comparables entre des séries temporelles de natures différentes. Cependant, cette normalisation peut être discutée, car certaines informations importantes, telles que la norme ou la moyenne des signaux, peuvent être perdues. Pour pallier ce problème, nous proposons d'ajouter la moyenne et l'écart-type des signaux au vecteur de paramètres qui les représentera par la suite.

Certaines méthodes de compression sont sensibles au bruit. Il est donc essentiel de filtrer ce bruit avant de procéder à la compression des données. Pour ce faire, nous utilisons un filtre passe-bas, dont la fréquence de coupure doit être soigneusement définie.

Filtrage passe-bas : Cette méthode permet de supprimer les hautes fréquences associées au bruit tout en conservant les composantes principales des séries temporelles (basses fréquences). Un filtre de Butterworth constitue une solution robuste pour cette tâche. Ce type de filtre offre la possibilité de définir une fréquence de coupure précise tout en ajustant la sélectivité selon les besoins.

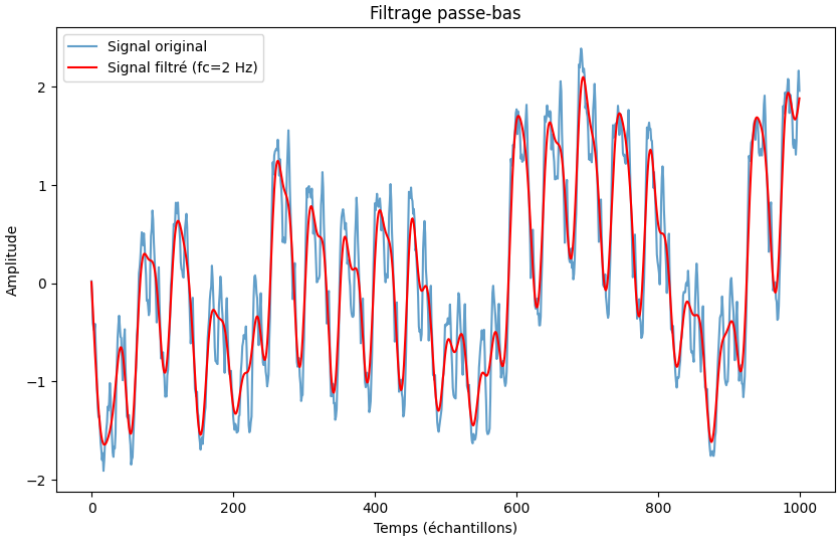


FIGURE 12 – Exemple de signal de consommation électrique filtré avec un filtre passe bas de fréquence de coupure de $2days^{-1}$.

Nous avons choisi une fréquence de coupure de $2\,\text{days}^{-1}$, car celle-ci permet de filtrer une grande partie du bruit tout en conservant une quantité significative d'information. En effet, la majorité de nos séries temporelles présentent une composante importante à $1\,\text{day}^{-1}$. Cela s'explique par le fait que le rythme de vie de la population est structuré par le cycle quotidien, tout comme les variations météorologiques, influencées par l'alternance jour/nuit.

4.2 Méthode de Compression

Dans cette sous-partie, nous explorons différentes méthodes de compression des signaux pour calculer les distances entre eux. Ces techniques permettent de transformer les séries temporelles en des vecteurs de petite dimension qui résument leurs principales caractéristiques. L'objectif est de mesurer la distance entre ces vecteurs de manière efficace et robuste, plutôt que de travailler directement sur les signaux bruts. Voici les justifications des trois méthodes retenues :

— **Transformée de Fourier Discrète (DFT) moyennée par bande :** Cette méthode est utilisée car une partie des séries temporelles présentent des comportements périodiques, ce qui rend pertinent l'analyse de leurs spectres fréquentiels. En moyennant les coefficients de la DFT par bande de fréquence, la méthode gagne en robustesse face à de légers décalages dans les fréquences dominantes des signaux. Cela permet une comparaison efficace tout en conservant les propriétés globales du spectre.

— **Modèle tendance + saisonnalité :** En complément de l'analyse fréquentielle, cette méthode introduit des informations supplémentaires en capturant la tendance et les déphasages dans les composantes saisonnières des signaux. Elle permet ainsi de représenter les séries temporelles de manière plus complète en intégrant des notions de direction globale et de synchronisation.

— **Autoencodeur (AE) :** Bien que la méthode tendance + saisonnalité soit perti-

nente, elle ne converge pas toujours de manière fiable sur certains signaux, ce qui peut entraîner des paramètres éloignés des valeurs optimales. Pour pallier cette limitation, nous utilisons un autoencodeur basé sur cette méthode. L'autoencodeur, bien qu'un peu moins précis, offre une meilleure robustesse en capturant des informations similaires tout en s'adaptant plus efficacement aux signaux complexes ou mal conditionnés.

Ces trois méthodes offrent une approche complémentaire et permettent de capturer différentes propriétés des séries temporelles, assurant ainsi une analyse robuste et cohérente des données.

4.2.1 DFT (Transformée de Fourier Discrète)

La DFT est très utilisée dans le traitement du signal, car elle permet d'étudier les composantes fréquentielles d'un signal. Nous allons rappeler sa définition [10].

Définition : Pour un signal discret $x[n]$ de longueur N, la Transformée de Fourier Discrète (DFT) est donnée par :

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn}, \quad k = 0, 1, \ldots, N-1, \tag{1}$$

où :
- $X[k]$ représente l'amplitude de la k-ième fréquence,
- N est la longueur du signal.

La DFT inverse est définie par :

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N}kn}. \tag{2}$$

Une fois les composantes fréquentielles obtenues, nous moyennons l'énergie par bande de fréquence afin d'obtenir un vecteur compressé représentatif de notre signal.

Pour ce faire, nous procédons comme suit :

1. Calculons l'énergie associée à chaque fréquence :

$$E[k] = |X[k]|^2, \quad k = 0, 1, \ldots, N-1, \tag{3}$$

   où $|X[k]|^2$ correspond à l'énergie associée à la k-ième fréquence.
2. Divisons le spectre en N bandes de largeur égales , notées $\{b_1, b_2, \ldots, b_B\}$.
3. Moyennons l'énergie dans chaque bande b_i :

$$E_b[i] = \frac{1}{|b_i|} \sum_{k \in b_i} E[k], \quad i = 1, 2, \ldots, N, \tag{4}$$

   où $|b_i|$ est le nombre de fréquences dans la bande b_i.

Le vecteur compressé résultant est constitué des $E_b[i]$, qui représentent l'énergie moyenne de chaque bande de fréquence.

Avantages de cette méthode :

— La compression réduit la dimensionnalité du signal tout en conservant les informations essentielles sur sa distribution fréquentielle.

— Cette représentation est robuste aux variations locales du signal, comme les bruits ou les petites perturbations.

Ainsi, chaque signal est réduit à un vecteur d'énergie moyenne par bande, qui peut être utilisé pour des comparaisons ou des analyses ultérieures.

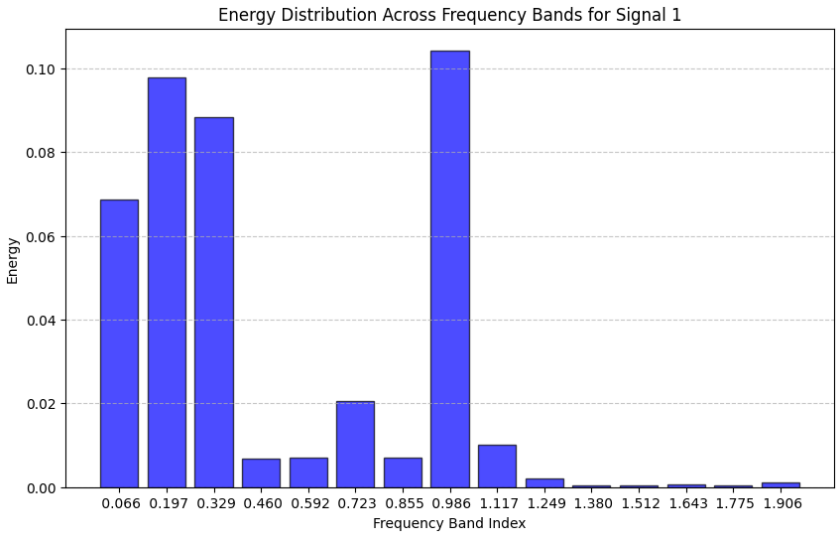


FIGURE 13 – Exemple de moyenne par bande de l'énergie fréquentielle de la consommation électrique.

Nous observons que l'énergie de notre signal se concentre majoritairement sur les première bandes et celle à $1Days^{-1}$.

4.2.2 Utiliser un modèle tendance et saisonnalité [8]

Afin de généraliser la notion de *smoothness*, une première approche consiste à ajuster un modèle à chaque série temporelle. Ce modèle doit capturer deux composantes fondamentales :

— **Tendance :** Une composante lente représentant les variations à long terme de la série. Cette tendance peut être modélisée à l'aide d'un polynôme de degré d ou d'une fonction lisse telle qu'une spline.

— **Saisonnalité :** Une composante oscillatoire qui décrit les variations périodiques de la série. Les fonctions sinusoïdales (cosinus avec déphasage) sont typiquement utilisées pour capturer ces dynamiques.

Ainsi, le modèle final pour une série temporelle $x(t)$ s'exprime comme suit :

$$x(t) = \sum_{i=0}^{d} a_i t^i + \sum_{j=1}^{N} \left[A_j \cos(2\pi f_j t + \phi_j) \right],$$

où :

— a_i sont les coefficients du polynôme représentant la tendance,
— A_j, f_j et ϕ_j correspondent respectivement à l'amplitude, la fréquence et le déphasage des composantes périodiques.

En ajustant ce modèle à chaque nœud du graphe, nous obtenons un ensemble de paramètres caractérisant chaque série temporelle. Ces paramètres peuvent ensuite être utilisés pour calculer des *features* au niveau des nœuds.

Cependant, dans la pratique, notre algorithme rencontre des difficultés de convergence. Cela s'explique par le fait que l'espace d'optimisation n'est pas nécessairement convexe. Par exemple, dans certains cas, l'algorithme peut utiliser un cosinus de très basse fréquence pour ajuster le signal, entraînant des résultats sous-optimaux.

Pour pallier ces limitations, nous avons adopté une méthode plus structurée en supposant que, pour des plages de fréquences restreintes, notre base de fonctions constitue une famille libre. Cette méthode suit les étapes suivantes :

— **Étape 1 :** Ajuster d'abord la tendance à l'aide d'un polynôme ou d'une spline.
— **Étape 2 :** Calculer la FFT du signal restant afin d'identifier les fréquences principales. Ces fréquences servent à initialiser les composantes périodiques du modèle.
— **Étape 3 :** Ajuster ensuite la partie saisonnalité, en affinant les amplitudes et les déphasages des composantes périodiques.

Cette approche améliore considérablement la fiabilité de l'algorithme tout en préservant une bonne précision.

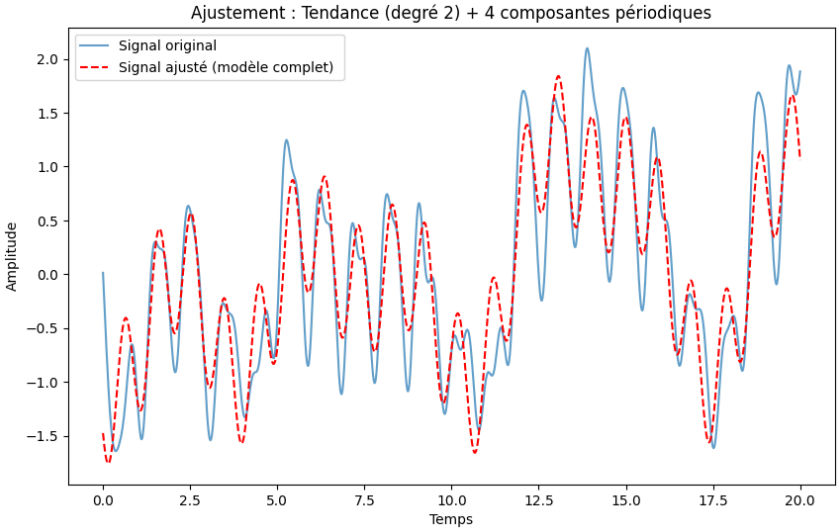


FIGURE 14 – Exemple d'ajustement du modèle *tendance+saisonnalité* à un signal de consommation électrique.

Comme illustré dans la figure 14, notre modèle parvient globalement à reproduire le signal d'entrée. Cependant, l'erreur quadratique reste significative, et le modèle ne converge pas toujours pour l'ensemble des signaux. Ces limitations soulignent la nécessité de développer une méthode plus robuste pour comparer les signaux.

C'est dans ce contexte que nous nous intéressons à l'utilisation des autoencodeurs. Ces derniers offrent une alternative capable de capturer des caractéristiques pertinentes des signaux, tout en garantissant une convergence plus fiable.

4.2.3 AE (AutoEncodeur)

Un autoencodeur est un modèle d'apprentissage profond supervisé utilisé pour la compression et la reconstruction de données. Il se compose de deux parties principales :

— **Encodeur** : Transforme les données d'entrée $x \in \mathbb{R}^d$ en une représentation latente $z \in \mathbb{R}^k$ avec $k < d$, via une fonction $h_\theta(x)$ paramétrée par θ.

— **Décodeur** : Reconstruit les données originales $\hat{x} \in \mathbb{R}^d$ à partir de la représentation latente z, via une fonction $g_\phi(z)$ paramétrée par ϕ.

La reconstruction est évaluée via une fonction de perte, par exemple, l'erreur quadratique moyenne :

$$\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2. \tag{5}$$

L'objectif est de minimiser cette perte afin que $\hat{x}$ soit aussi proche que possible de x :

$$\min_{\theta,\phi} \mathbb{E}_{x \sim p_{\text{data}}}[\mathcal{L}(x, g_\phi(h_\theta(x)))]. \tag{6}$$

Voilà la strucure d'un AE sur un schéma claire :

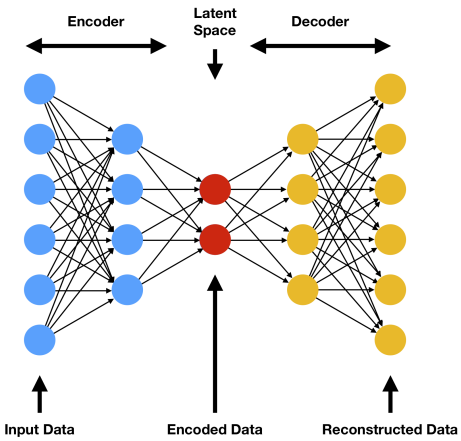


FIGURE 15 – image de la structure d'un AE. [lien image]

Dans l'état actuel, l'espace latent généré par l'autoencodeur (AE) présente un désordre notable. Deux signaux perçus comme similaires peuvent se retrouver éloignés dans cet espace, tandis que des signaux très différents peuvent apparaître proches. Cette configuration constitue une limitation majeure, car l'objectif de l'AE est précisément de mesurer les distances entre signaux de manière cohérente.

Pour surmonter cette contrainte, il est nécessaire de redéfinir la structure de l'autoencodeur. Nous avons ainsi intégré une fonction de perte composée de deux termes principaux : la perte de reconstruction et la perte de distance. La perte de reconstruction garantit que le décodeur peut reproduire avec précision le signal d'entrée à partir de sa représentation latente. La perte de distance, quant à elle, s'assure que les distances dans l'espace latent reflètent fidèlement les distances entre les paramètres des signaux d'origine. Cette fonction de perte est définie comme suit [7, 3] :

$$\mathcal{L} = \alpha \mathcal{L}_{\text{reconstruction}} + \beta \mathcal{L}_{\text{distance}},$$

où α et β sont des hyperparamètres contrôlant l'importance relative de chaque terme.

Plus précisément, cette fonction de perte s'écrit :

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\|s_i(t) - \hat{s}_i(t)\|^2 + \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j\|_2 - \|\mathbf{z}_i - \mathbf{z}_j\|_2\right)^2,$$

avec :

— $\boldsymbol{\theta}_i$ et $\boldsymbol{\theta}_j$ les paramètres caractéristiques des signaux s_i et s_j.

— $\mathbf{z}_i$ et $\mathbf{z}_j$ les vecteurs dans l'espace latent correspondant aux signaux s_i et s_j.

Dans cette formulation, la première composante $\mathcal{L}_{\text{reconstruction}}$ veille à la fidélité entre le signal d'entrée et sa reconstruction. La seconde composante $\mathcal{L}_{\text{distance}}$ mesure l'écart entre les distances dans l'espace des paramètres et celles dans l'espace latent. Cette dernière se base sur la méthode k-NN, comme détaillé dans [3], pour comparer les matrices de distances des signaux.

Dans la pratique, l'apprentissage de l'AE s'effectue par lots (*batch size*), ce qui permet une estimation robuste des distances moyennes. Cette approche a montré de bons résultats pour préserver les métriques de distance dans l'espace latent, et elle constitue actuellement la méthode la plus performante que nous avons testée. Bien que des alternatives soient en cours d'exploration, elles ne seront pas détaillées dans ce rapport.

Cependant, il est important de noter que cette méthode nécessite une connaissance *a priori* d'un modèle adapté aux données réelles, car cette notion de distance repose sur ce modèle. Par conséquent, nous avons limité l'application de notre autoencodeur aux données de consommation électrique. Par manque de temps, nous n'avons pas développé de modèle équivalent pour les autres séries temporelles. Nous avons choisi le modèle tendance + saisonnalité pour définir notre notion de distance.

Pour vérifier le bon fonctionnement de notre autoencodeur, nous avons effectué deux types de tests :

1. **Reconstruction du signal :** Vérifier que l'autoencodeur parvient à reproduire un signal réel avec précision.
2. **Respect de la notion de distance :** Générer une séquence de signaux en faisant varier progressivement un paramètre (les autres restant fixés) et observer si les distances dans l'espace latent respectent cette progression.

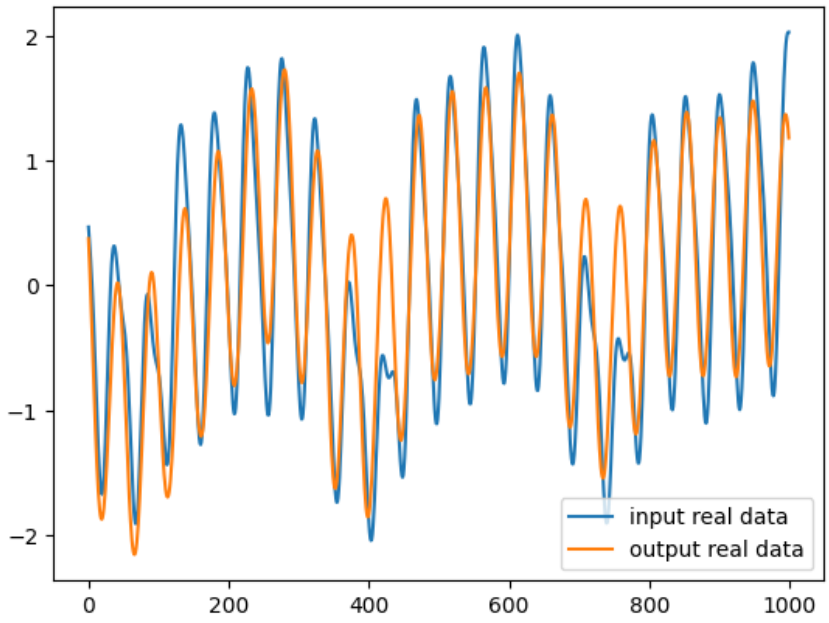


FIGURE 16 – Reconstruction d'un signal de consommation électrique par l'autoencodeur.

Sur la figure 16, on observe que l'autoencodeur reproduit fidèlement le signal d'entrée, confirmant la qualité de la reconstruction.

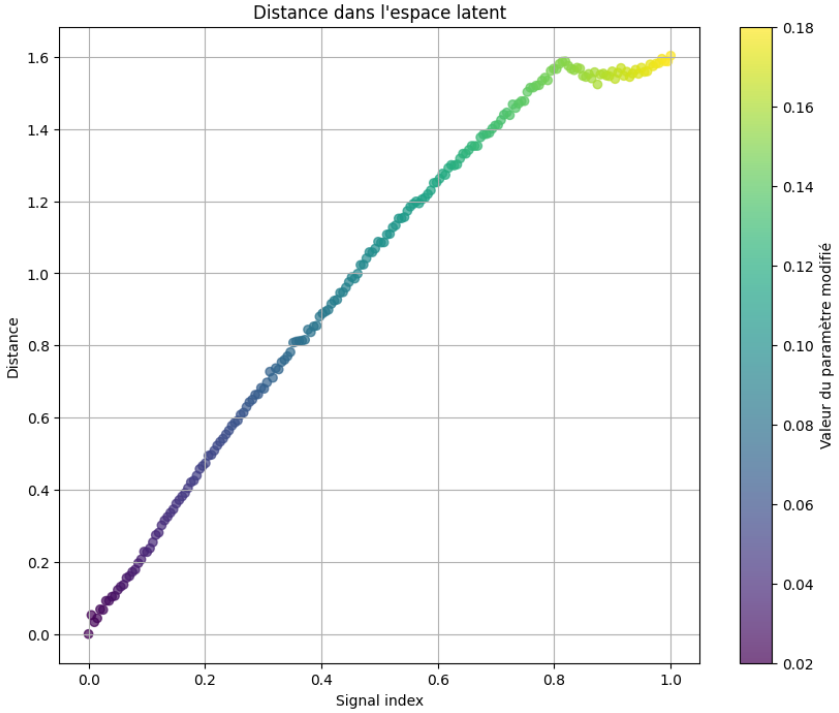


FIGURE 17 – Visualisation des distances dans l'espace latent. Une séquence de signaux a été générée en faisant varier progressivement un paramètre, et la distance de chaque signal par rapport au premier a été tracée.

La figure 17 montre que les distances dans l'espace latent respectent globalement les

variations progressives des paramètres des signaux. Toutefois, à l'extrémité de la séquence, les distances deviennent moins cohérentes, indiquant une légère perte de précision de la métrique.

Ces résultats confirment que notre autoencodeur est capable de capturer les relations entre signaux de manière fiable tout en conservant une structure cohérente dans l'espace latent.

4.3 Généralisation de plusieurs features

Une fois que chaque série temporelle est résumée par ses paramètres (tendance et périodicité), nous pouvons calculer la smoothness sur le graphe en utilisant ces paramètres comme des caractéristiques scalaires des nœuds.

4.3.1 Généralisation de la smoothness

Définition : Pour un graphe $G = (V, E)$ où chaque nœud i est associé à un vecteur de paramètres θ_i, la smoothness généralisée peut être définie comme :

$$\text{Smoothness généralisée} = \sum_{(i,j)\in E} w_{ij}\|\theta_i - \theta_j\|^2,$$

où :

- $\mathbf{p}_i$ est le vecteur des paramètres du modèle ajusté pour le nœud i,
- w_{ij} est le poids de l'arête (i, j).

Ce calcul mesure la cohérence des dynamiques des séries temporelles entre nœuds connectés. Si les séries temporelles associées aux nœuds voisins ont des paramètres similaires, la smoothness sera faible, ce qui reflète une grande homogénéité temporelle sur le graphe.

Une autre manière de voir la formule est de faire la somme des smoothness de chaque paramètre :

$$\text{Smoothness généralisée} = \sum_{k=1}^{N} S_k = tr(\theta^\top \mathcal{L}\theta) = \sum_{k=1}^{N} \sum_{(i,j)\in E} w_{ij}\|\theta_{i,1} - \theta_{j,1}\|^2,$$

où :

- S_k est la contribution à la smoothness pour le k-ième paramètre,
- w_{ij} est le poids de l'arête (i, j), qui mesure la proximité ou l'importance de la relation entre les nœuds i et j,
- $\mathcal{L}$ est le laplacien du graphe.
- $\|\theta_{i,k} - \theta_{j,k}\|^2$ est la différence quadratique entre le k-ième paramètre des nœuds i et j.

4.3.2 Généralisation de la local smoothness

Définition : Pour chaque nœud i, la *local smoothness* peut également être définie en comparant $\mathbf{p}_i$ à ses voisins immédiats :

$$\lambda(n) = \frac{\sum_{j \in \mathcal{N}(n)} w_{ij} \|\theta_i - \theta_j\|^2}{\sum_{k=0}^{N} \|\theta_k\|^2},$$

où $\mathcal{N}(n)$ est l'ensemble des voisins du nœud n.
Cette mesure fournit une granularité plus fine, permettant d'identifier des zones spécifiques du graphe où les séries temporelles diffèrent significativement, sans prendre en compte l'ensemble global du graphe.

4.3.3 Généralisation de la bandwith

Cette notion est légèrement plus compliqué à généraliser que les précédente. Revenons en à la notion physique de la bandwith, cette dernière est sensé mesurer la dispersion fréquentielle de chaque nœud. Pour ce faire, il faut utiliser normalement des distribution vertex-frequency qui respecte certaines propriétés mathématiques. Cette distribution doit nous permettre de calculer la fréquence moyenne du signal à travers le graphe. Nous en venons donc à la définition suivante :

Définition : Bandwith (bande passante) :
Largeur de bande généralisée d'un signal f évoluant à travers le temps :

$$B = \sqrt{\frac{\sum_{k=1}^{N} (\lambda_k - \lambda(n))^2 ||\hat{f}_k * U_k(n)||^2}{\sum_{k=1}^{N} ||\hat{f}_k * U_k(n)||^2}}$$

Si le dénominateur est nul :

$$B = 0$$

- avec $\lambda(n)$ la local smoothness généralisé.
- $U_k(n)$ correspond à la n-ème termes du k-eme vecteur propre
- f correspond à un signal sur graphe évoluant dans le temps, c'est à dire qu'il y a une séries temporelles par noeuds. Donc la valeurs de f à un noueds données à un temps données se note $f(n,t)$. Afin de procéder à la GFT de f nous calculons à temps fixer la GFT et obtenons $\hat{f}(k,t)$, avec k l'indice de la fréquence. Ainsi, $||\hat{f}_k|| = \frac{\sqrt{\sum_{t=1}^{T} (\hat{f}(k,t))^2}}{T}$
  Or $U_k(n)$ est un scalaire et donc $||\hat{f}_k * U_k(n)|| = |U_k(n)| * \sqrt{\frac{\sum_{t=1}^{T} (\hat{f}(k,t))^2}{T}}$

4.4 Algorigramme

Afin d'avoir une vision claire et synthétique de la démarche mise en œuvre dans cette partie, nous présentons ci-dessous un algorigramme. Cet algorigramme détaille les différentes étapes, de l'entrée des données brutes jusqu'à l'analyse des *features*, en passant par

la construction et la structuration des graphes.

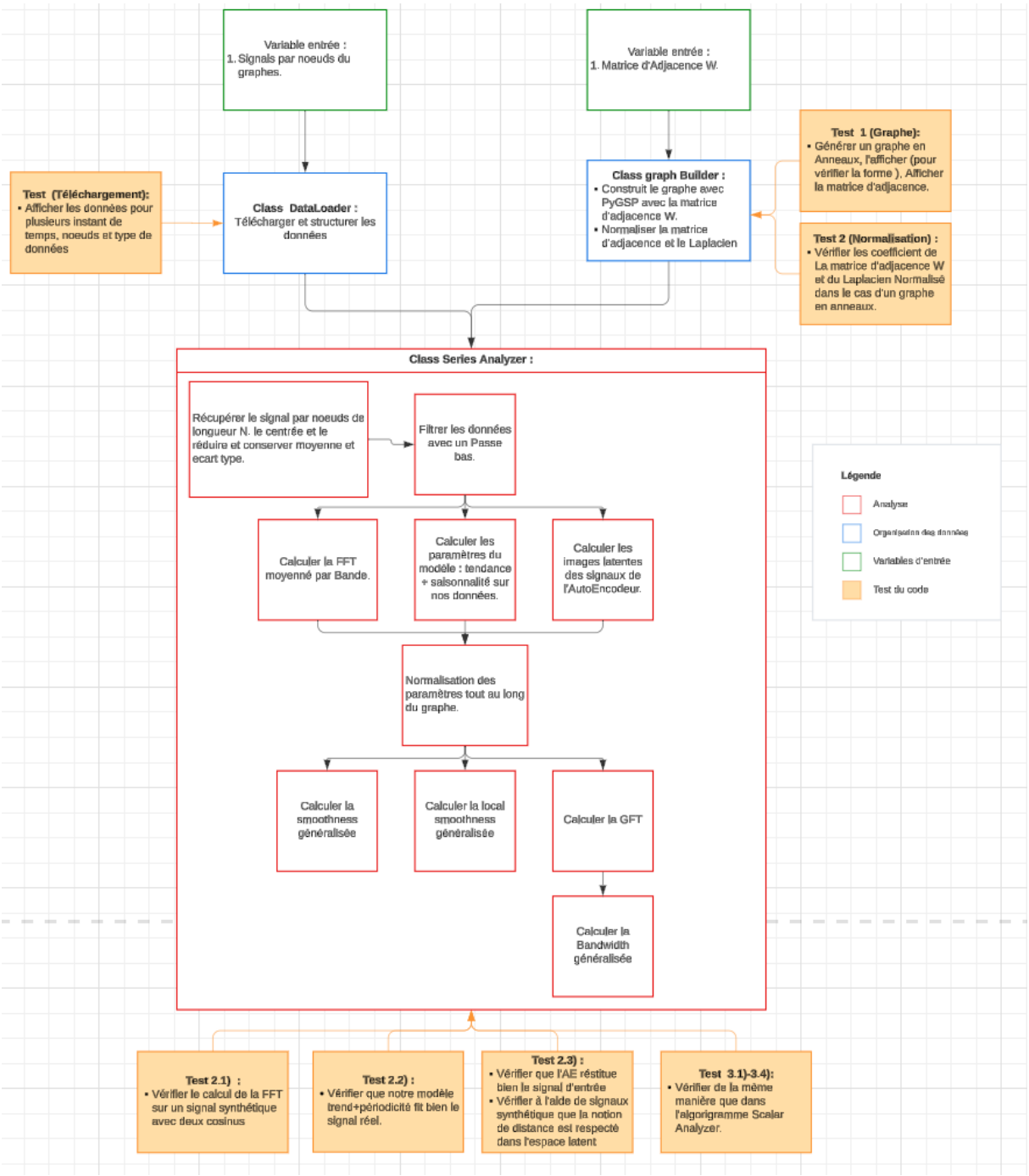


FIGURE 18 – Algorigramme des étapes principales de l'analyse des séries temporelles et des *features* appliquées aux graphes.

Les principales étapes sont décrites comme suit :

— **Variables d'entrée :** Les données brutes consistent en deux types principaux :

1. Les signaux temporels associés aux nœuds des graphes.
2. La matrice d'adjacence W, qui définit la connectivité des graphes.

— **Classe `DataLoader` :** Cette étape concerne le téléchargement et la structuration des données pour préparer les signaux bruts à l'analyse.
— **Classe `Graph Builder` :**
  — Construction des graphes à partir de la matrice d'adjacence en utilisant `PyGSP`.
  — Normalisation des matrices d'adjacence et du Laplacien.
  — Application d'un filtrage pour rendre les graphes plus parcimonieux.
— **Classe `Series Analyzer` :**
  — Analyse des signaux associés aux graphes :
    1. Extraction des signaux par nœud sur une longueur N.
    2. Filtrage des données à l'aide d'un filtre passe-bas de Butterworth.
  — Calcul des paramètres spécifiques aux signaux :
    1. Moyenne spectrale par bande via la FFT.
    2. Décomposition en tendance et périodicité.
    3. Calcul des paramètres dans l'espace latent d'un auto-encodeur.
  — Analyse des *features* :
    1. Calcul global et local de la *smoothness*.
    2. Transformée de Fourier sur graphe (GFT).
    3. Calcul global et local de la *bandwidth*.

4.5 Résultats

Dans cette sous partie, nous allons procéder à une analyse similaire à celle de la partie précédente. Nous allons dans un premier temps afficher les features sur leur densité de probabilité calculer à l'aide de la méthode de monte-carlo sur les données de consommation électrique.

Nous rappelons que par soucis de lisibilité nous avons attribué des indices aux graphes de l'article [2] :
— Graphe 0 : Correlation
— Graphe 1 : Distspline
— Graphe 2 : Distpline2
— Graphe 3 : DTW
— Graphe 4 : GL3SR
— Graphe 5 : Precision
— Graphe 6 : Space

4.5.1 Résultats features

Afin d'évaluer les caractéristiques qui définissent la qualité d'un graphe, nous avons calculé les *features* de *smoothness* et de *bandwidth*. Cependant, ces *features* classiques ne sont définies que pour des signaux scalaires par nœud. Dans notre cas, chaque nœud est associé à une série temporelle, rendant nécessaire une généralisation de ces notions.

Pour la *smoothness*, nous avons généralisé la notion de distance entre signaux en utilisant trois méthodes distinctes :

— Compression par Transformée de Fourier Discrète (FFT) : les signaux sont projetés dans l'espace fréquentiel, et leur énergie est moyennée par bande de fréquence. La distance entre les spectres est ensuite calculée.
— Ajustement d'un modèle tendance + saisonnalité : nous avons ajusté chaque signal avec un modèle combinant une tendance et des composantes périodiques, puis calculé la distance entre les paramètres ajustés.
— Projection dans l'espace latent d'un autoencodeur (AE) : les signaux sont projetés dans un espace latent appris, et la distance est mesurée dans cet espace.

Pour la *bandwidth*, nous avons utilisé la décomposition spectrale pour calculer une mesure agrégée sur chaque nœud. Nous affichons également l'écart-type de la *bandwidth* sur l'ensemble des nœuds afin de capturer la variabilité des graphes.

Nous présentons ci-dessous les figures correspondantes, qui illustrent les valeurs des *features* pour chaque graphe, superposées à leurs densités de probabilités respectives calculées sur une période de 21 jours. Ces résultats fournissent des informations précieuses sur la structure des graphes et leurs interactions avec les séries temporelles de consommation électrique.

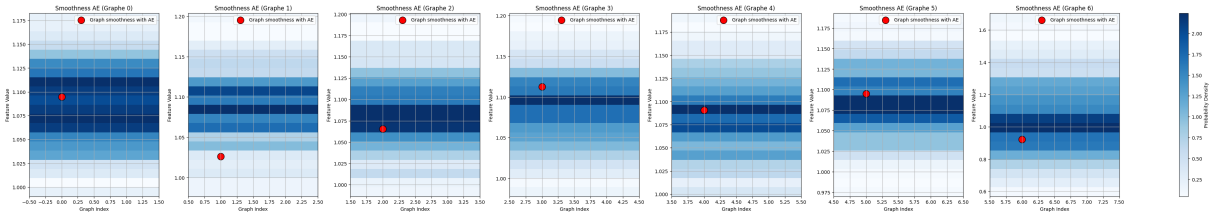


FIGURE 19 – *Smoothness* calculée dans l'espace latent de l'autoencodeur (AE) pour les graphes, superposée à leur densité de probabilité respective.

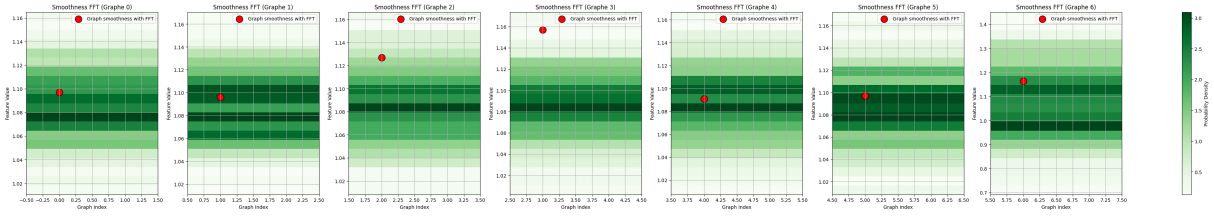


FIGURE 20 – *Smoothness* calculée à partir des spectres FFT des séries temporelles des graphes, superposée à leur densité de probabilité respective.

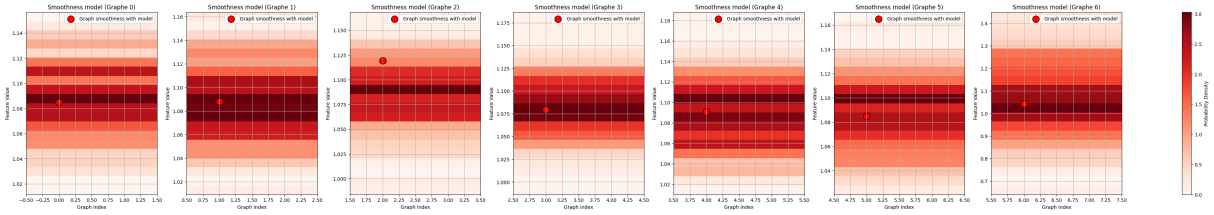


FIGURE 21 – *Smoothness* calculée à partir des paramètres ajustés avec le modèle tendance + saisonnalité, superposée à leur densité de probabilité respective.

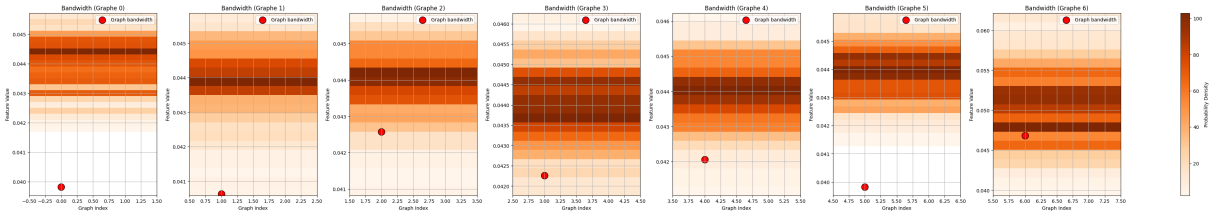


FIGURE 22 – *Bandwidth* calculée pour les graphes, superposée à leur densité de probabilité respective.

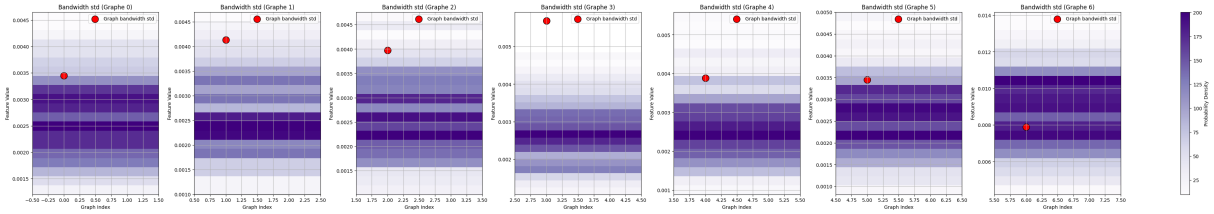


FIGURE 23 – Écart-type de la *bandwidth* calculée pour les graphes, superposée à leur densité de probabilité respective.

4.5.2 Analyse des résultats

Les figures 19, 20, 21, 22 et 23 présentent respectivement les valeurs des *features* calculées pour différents graphes, en utilisant des méthodes complémentaires : la *smoothness* via l'espace latent d'un autoencodeur (AE), les spectres FFT et un modèle tendance + saisonnalité, ainsi que la *bandwidth* et son écart-type. Ces figures permettent d'évaluer si les connexions présentes dans les graphes étudiés traduisent une structure logique, ou si elles sont similaires à des graphes aléatoires.

Graphe 0 (Corrélation) Le graphe 0, basé sur les corrélations directes entre les séries temporelles réduites, montre des résultats qui suggèrent des connexions proches d'un comportement aléatoire :

- Toutes les méthodes de calcul de la *smoothness* (AE, FFT, modèle) montrent des valeurs dans la zone de haute densité des graphes aléatoires. Cela indique que les connexions établies dans ce graphe ne traduisent pas de structure significative.
- Cette observation est attendue, car la corrélation simple est sensible au bruit et aux déphasages entre les signaux, ce qui limite sa capacité à révéler des relations structurelles pertinentes.

Graphes 1 (Distspline) et 2 (Distspline2) Ces graphes, construits à partir des distances entre splines, montrent des comportements différents malgré une méthode similaire :

- Le graphe 1 présente une *smoothness* calculée via AE largement en dehors de la zone de haute densité, indiquant que ce graphe capture une structure cohérente avec les séries temporelles. Cependant, les autres méthodes (FFT et modèle) placent ses valeurs dans les zones de densité élevée, ce qui laisse penser que cette cohérence est limitée à des aspects globaux du signal.

— À l'inverse, le graphe 2 montre des valeurs de *smoothness* hors de la zone de densité uniquement pour les méthodes FFT et tendance + saisonnalité, mais pas pour l'AE. Cela peut indiquer que ce graphe est plus sensible aux relations spectrales ou temporelles spécifiques.

Graphe 3 (DTW) La méthode DTW, qui aligne dynamiquement les signaux, montre un comportement contrasté :

— La méthode FFT place la *smoothness* du graphe 3 largement hors de la zone de haute densité, suggérant que DTW capture des relations cohérentes sur le plan fréquentiel.

— Cependant, les méthodes AE et tendance + saisonnalité montrent des valeurs dans la zone de haute densité, ce qui pourrait indiquer que DTW ne parvient pas à aligner suffisamment bien les signaux pour révéler des relations structurelles fortes selon ces métriques.

Graphe 4 (GL3SR) Basé sur des hypothèses de lissage et de bande limitée, ce graphe montre une cohérence marquée :

— Toutes les méthodes placent les valeurs de *smoothness* dans la zone de haute densité des graphes aléatoires. Cela peut paraître contre-intuitif, mais cela pourrait indiquer que la construction du graphe GL3SR est fortement influencée par des hypothèses qui ne sont pas capturées par ces métriques.

Graphe 5 (Précision) Construit à partir de l'inverse de la matrice de covariance, ce graphe montre des connexions conditionnelles modélisées de manière approximative :

— Toutes les méthodes de calcul placent ses valeurs de *smoothness* dans la zone de haute densité. Cela suggère que les relations conditionnelles modélisées par ce graphe ne traduisent pas des structures significatives dans les séries temporelles.

Graphe 6 (Espace) Le graphe basé sur la proximité géographique des nœuds montre des connexions cohérentes :

— Les méthodes AE et FFT montrent des valeurs légèrement en dehors de la zone de haute densité, mais pas de manière significative, tandis que la méthode tendance + saisonnalité montre une valeur dans la zone de haute densité.

— Ces résultats indiquent que les connexions géographiques capturent des relations locales, mais que cette structure n'est pas systématiquement révélée par toutes les méthodes.

Bandwidth et écart-type Les figures 22 et 23 présentent les valeurs de *bandwidth* et son écart-type :

— Toutes les *bandwidth* se situent sous la zone de haute densité des graphes aléatoires, sauf pour le graphe 6 (Espace). Cela indique que les graphes étudiés, à l'exception du graphe géographique, traduisent des signaux peu dispersés dans le domaine fréquentiel.

— L'écart-type de la *bandwidth*, représentant la variabilité entre les nœuds, se situe au-dessus des zones de haute densité pour tous les graphes, sauf le graphe 6. Cela

montre que la dispersion fréquentielle varie fortement d'un nœud à l'autre, sauf dans le cas des connexions géographiques.

Les figures présentées ne permettent pas d'identifier clairement des caractéristiques significatives des graphes pour évaluer leur qualité en termes de performances prédictives des GNN. En particulier, le graphe GL3SR, qui montre les meilleures performances, ne présente pas de valeurs distinctives pour les *features* calculées. Ce constat suggère que les caractéristiques essentielles des graphes se trouvent ailleurs.

Plusieurs hypothèses peuvent expliquer ces résultats :

— **Échelles temporelles :** Les *features* calculées sur une échelle de temps fixe pourraient ne pas capturer les dynamiques pertinentes. D'autres échelles temporelles, comme mensuelles ou annuelles, pourraient être plus appropriées.

— **Séries temporelles utilisées :** Les données de consommation électrique, bien qu'importantes, ne sont peut-être pas suffisantes. Des séries comme les données météorologiques pourraient mieux refléter les relations structurelles.

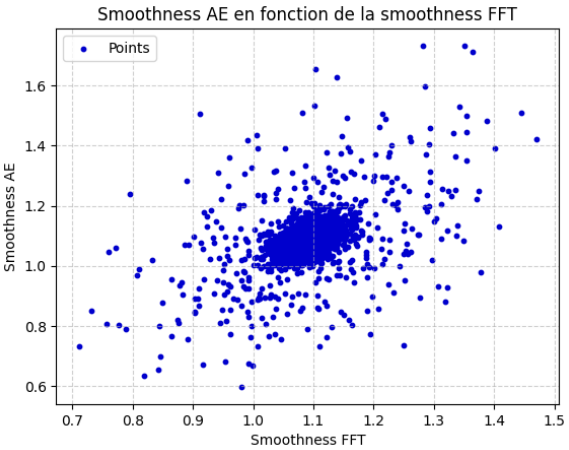
— **Normalisation :** La normalisation des graphes pourrait altérer des propriétés cruciales, notamment pour le graphe GL3SR, initialement caractérisé par des poids très faibles.

— **Autres *features* :** Des métriques non explorées, comme les propriétés spectrales ou la connectivité globale, pourraient mieux expliquer les performances des GNN.

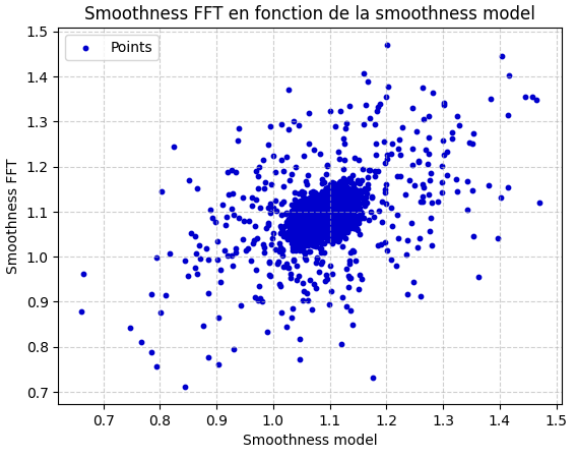
Pour aller plus loin, il serait pertinent d'explorer différentes échelles de temps, d'autres séries temporelles et des approches alternatives de calcul des *features*. Une analyse approfondie de l'impact de la normalisation des graphes pourrait également apporter des éclairages nouveaux.

4.5.3 Corrélation entre les *features*

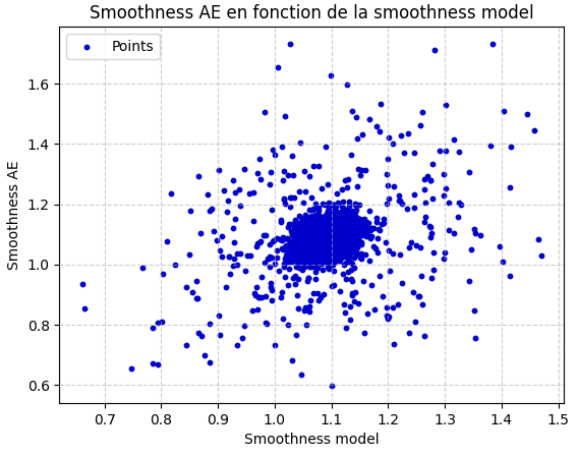
Afin de compléter notre analyse, nous étudions la corrélation entre les différentes *smoothness* calculées à l'aide des trois méthodes (AE, FFT et tendance + saisonnalité). Cela nous permettra d'identifier les relations intrinsèques entre ces trois *features*.



(a) Corrélation entre la *smoothness* calculée par l'autoencodeur (AE) et celle calculée à partir des spectres FFT.



(b) Corrélation entre la *smoothness* calculée à partir des spectres FFT et celle calculée par le modèle tendance + saisonnalité.



(c) Corrélation entre la *smoothness* calculée par l'autoencodeur (AE) et celle calculée par le modèle tendance + saisonnalité.

FIGURE 24 – Corrélations entre les différentes méthodes de calcul de la *smoothness*. Chaque sous-figure représente une combinaison spécifique de méthodes.

Sur ces figures, nous observons que les trois *smoothness* présentent des corrélations faibles à modérées entre elles. En particulier, la *smoothness* calculée par l'AE est plus fortement corrélée avec celle obtenue via la FFT (figure 24a), tandis que les corrélations entre la *smoothness* du modèle tendance + saisonnalité et celles des deux autres méthodes sont légèrement plus faibles (figure 24b, figure 24c).

Les corrélations observées suggèrent que les trois méthodes capturent en partie des informations communes, probablement liées aux propriétés fréquentielles des signaux. Cela est cohérent avec la nature des calculs effectués :

— La méthode AE extrait des représentations dans un espace latent, qui semble sensible aux structures fréquentes ou périodiques des signaux.

— La méthode FFT calcule directement les amplitudes des fréquences, fournissant

une mesure explicite de la répartition de l'énergie spectrale.
— La méthode tendance + saisonnalité, bien que basée sur des paramètres ajustés (amplitude, fréquence et phase), capture également des informations liées à la fréquence globale des signaux.

Cependant, les corrélations ne sont pas parfaites, ce qui peut s'expliquer par les différences fondamentales entre les trois approches. Par exemple :
— L'AE intègre une compression des signaux qui pourrait perdre des détails spectraux fins.
— La FFT, bien qu'exhaustive en termes de spectre, ne prend pas en compte les variations locales ou les déphasages entre les signaux.
— Le modèle tendance + saisonnalité est sensible à la qualité de l'ajustement des paramètres, ce qui pourrait introduire une variabilité supplémentaire.

En conclusion, bien que les trois *smoothness* soient liées, leurs différences méthodologiques permettent de fournir des perspectives complémentaires sur la qualité des graphes étudiés.

5 Conclusion

Dans ce travail, nous avons exploré différentes approches pour analyser et comprendre les caractéristiques des graphes dans le contexte de prédiction avec des réseaux de neurones de graphes (GNN). Dans un premier temps, nous avons étudié des *features* calculées sur des signaux bruts, tels que la *smoothness* et la *bandwidth*, et avons démontré leur capacité à capturer des structures non aléatoires dans les données. Cette approche a permis de mettre en lumière des connexions cohérentes dans les graphes et de poser les bases pour une analyse plus approfondie. Cependant, nous avons constaté que ces *features* présentent certaines limitations, notamment leur incapacité à capturer des informations complexes lorsque les signaux bruts sont trop bruités, déphasés ou contractés/dilatés.

Pour pallier ces limitations, nous avons introduit des méthodes de compression, telles que la transformée de Fourier discrète (DFT) moyennée par bande, les modèles basés sur la tendance et la saisonnalité, et enfin, les autoencodeurs (AE). Ces approches ont permis de réduire la dimensionnalité des signaux tout en conservant leurs caractéristiques essentielles, rendant ainsi l'analyse des *features* plus robuste et fiable. En particulier, l'autoencodeur a permis de modéliser des signaux de manière plus cohérente dans l'espace latent, bien que certaines limitations subsistent dans les cas extrêmes.

Dans l'ensemble, nos travaux ont permis de poser des fondations pour l'analyse des graphes en combinant des approches basées sur des *features* globales et des techniques de compression. Cependant, plusieurs axes d'amélioration restent à explorer pour renforcer et étendre cette méthodologie.

Perspectives et travaux futurs

Pour aller plus loin, plusieurs directions de recherche peuvent être envisagées :
— **Utilisation de modèles additifs généralisés (GAM) :** Appliquer des GAM sur les séries temporelles de chaque nœud et extraire leurs paramètres comme nouvelles *features*. Ces paramètres permettraient d'étudier si les graphes connectent des nœuds ayant des dynamiques similaires dans leurs prédictions.

— **Analyse multi-échelle :** Étudier les *features* à différentes échelles, en les appliquant non seulement sur les signaux agrégés des graphes, mais également sur chaque série temporelle individuelle associée aux nœuds. Cela permettrait de mieux comprendre les variations locales des *features*.
— **Prise en compte d'informations non temporelles :** Intégrer des données non signalétiques, telles que les dates, les vacances scolaires ou d'autres variables explicatives, pour enrichir l'analyse et évaluer leur impact sur les *features* calculées.
— **Exploration de nouvelles *features* :** Introduire des métriques supplémentaires, comme la *band-limitedness*, qui mesure la parcimonie spectrale d'un signal dans le domaine de Fourier (GFT). Ces *features* pourraient fournir de nouvelles perspectives sur la structure et l'informativité des graphes.
— **Corrélation entre *features* et performances du GNN :** Une fois les *features* identifiées, il sera crucial d'établir leur lien avec les performances du GNN. Pour ce faire, nous pourrions générer une séquence de graphes en dégradant progressivement un graphe initial performant, puis mesurer simultanément la performance du GNN et la valeur des *features*. Cette approche permettrait d'étudier la corrélation entre la dégradation des performances et l'évolution des *features*, en s'appuyant éventuellement sur des méthodes non linéaires pour capturer des relations complexes.
— **Optimisation des graphes :** Une fois les *features* les plus importantes identifiées, un dernier objectif consisterait à développer des algorithmes capables de construire directement des graphes ayant les caractéristiques optimales. Cela permettrait d'induire des performances maximales pour les GNN en exploitant pleinement les propriétés des données.

En conclusion, ce travail ouvre des perspectives intéressantes pour l'analyse des graphes dans le contexte des réseaux de neurones. En combinant des approches analytiques rigoureuses, des méthodes de compression adaptées et des techniques d'optimisation, il sera possible d'approfondir notre compréhension des liens entre les propriétés des graphes et les performances des GNN. Ces avancées permettront non seulement de concevoir des graphes optimaux, mais également de mieux exploiter les données complexes et multi-échelles rencontrées dans des applications variées.

Références

[1] Eloi Campagne, Yvenn Amara-Ouali, Yannig Goude, and Argyris Kalogeratos. Forecasting net load in france : The edf data challenge. *EDF Data Challenge*, 2024.

[2] Eloi Campagne, Yvenn Amara-Ouali, Yannig Goude, and Argyris Kalogeratos. Leveraging graph neural networks to forecast electricity consumption. *arXiv preprint arXiv :2408.17366*, 2024.

[3] Nutan Chen, Patrick van der Smagt, and Botond Cseke. Local distance preserving auto-encoders using continuous knn graphs. In *ICML Workshop on Topology, Algebra, and Geometry in Machine Learning*, 2022.

[4] Miloš Daković, Ljubiša Stanković, and Ervin Sejdić. Local smoothness of graph signals. *Mathematical Problems in Engineering*, 2019 :1–14, 2019. Hindawi, Published : 8 April 2019.

[5] PyGSP Developers. Pygsp : Graph signal processing toolbox, 2024. Accessed : 2024-01-26.

[6] Pierre Humbert, Batiste Le Bars, Laurent Oudre, Argyris Kalogeratos, and Nicolas Vayatis. Learning laplacian matrix from graph signals with sparse spectral representation. *Journal of Machine Learning Research*, 22 :1–47, 2021. Submitted : 11/19 ; Revised : 4/21 ; Published : 10/21.

[7] Renato Miyagusuku and Koichi Ozaki. Distance invariant sparse autoencoder for wireless signal strength mapping. *arXiv preprint arXiv :2010.15347*, 2020.

[8] Laurent Oudre. Lecture 3 : Models and representation learning, 2023. Accessed : 2024-01-26.

[9] David I. Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs : Extending high-dimensional data analysis to networks and other irregular domains. *arXiv preprint arXiv :1211.0053*, 2013.

[10] John Smith. An introduction to the discrete fourier transform. *All About Circuits*, 2015.

6 Annexe

6.1 démonstration de l'action du laplacien combinatoire

Preuve :

Soit un graphe pondéré non orienté avec un Laplacien combinatoire L défini par :

$$L = D - W$$

Considérons maintenant un signal f défini sur les sommets du graphe, représenté par un vecteur $f \in \mathbb{R}^N$, où chaque composante $f(i)$ est la valeur du signal au sommet i.

L'action du Laplacien L sur le signal f est donnée par le produit matrice-vecteur Lf. La i-ème composante de ce produit, notée $(Lf)_i$, est :

$$(Lf)_i = \sum_j L_{i,j} f(j)$$

En substituant $L = D - W$, nous obtenons :

$$(Lf)_i = \sum_j (D_{i,j} - W_{i,j}) f(j)$$

Puisque D est une matrice diagonale, seuls les éléments diagonaux $D_{i,i}$ sont non nuls, donc l'expression devient :

$$(Lf)_i = D_{i,i} f(i) - \sum_j W_{i,j} f(j)$$

Rappelons que $D_{i,i} = \sum_j W_{i,j}$, ce qui nous permet de réécrire $(Lf)_i$ comme suit :

$$(Lf)_i = f(i) \sum_j W_{i,j} - \sum_j W_{i,j} f(j)$$

En factorisant $f(i)$ dans le premier terme, nous obtenons :

$$(Lf)_i = \sum_j W_{i,j}(f(i) - f(j))$$

Ainsi, l'action du Laplacien sur le signal f au sommet i est donnée par :

$$(Lf)_i = \sum_{j \in \mathcal{N}_i} W_{i,j}(f(i) - f(j))$$

où $\mathcal{N}_i$ représente l'ensemble des sommets connectés au sommet i. □

6.2 Démonstration GFT

Preuve (GFT) :

La transformée de Fourier sur graphe repose sur la propriété fondamentale de la décomposition spectrale du Laplacien. Comme L est une matrice symétrique et semi-définie positive, elle possède une décomposition spectrale sous la forme :

$$L = U \Lambda U^\top,$$

où :
- $U = [u_0, u_1, \ldots, u_{N-1}]$ est une matrice dont les colonnes sont les vecteurs propres orthonormés de L.
- $\Lambda = \text{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{N-1})$ est une matrice diagonale contenant les valeurs propres $\lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{N-1}$ de L.

Pour un signal $f \in \mathbb{R}^N$, nous pouvons exprimer f dans la base des vecteurs propres de L :

$$f = \sum_{l=0}^{N-1} \langle f, u_l \rangle u_l = \sum_{l=0}^{N-1} \hat{f}(\lambda_l) u_l,$$

où chaque coefficient $\hat{f}(\lambda_l)$ est donné par la projection de f sur le vecteur propre u_l, soit :

$$\hat{f} = U^\top f.$$

Cette transformation $\hat{f} = U^\top f$ fournit les coefficients de f dans le domaine spectral du graphe. La transformation inverse, $f = U\hat{f}$, permet de reconstruire f en combinant ses composantes dans le domaine des vecteurs propres du Laplacien. □

6.3 Démonstration du théorème de Courant Fischer

Nous allons démontrer ce théorème par réccurence.

Initialisation : la Plus Petite Valeur Propre λ_1

Pour déterminer la plus petite valeur propre λ_1, nous considérons le problème suivant :

$$\lambda_1 = \min_{\substack{x \in \mathbb{R}^n \\ \|x\|=1}} x^\top A x.$$

— Puisque A est symétrique, elle possède une base orthonormée de vecteurs propres $v_1, v_2, \ldots, v_n$ associés aux valeurs propres $\lambda_1, \lambda_2, \ldots, \lambda_n$.
— Pour tout vecteur $x \in \mathbb{R}^n$, on peut écrire x comme une combinaison linéaire des vecteurs propres de A :

$$x = \sum_{i=1}^{n} \alpha_i v_i,$$

où $\alpha_i = v_i^\top x$ sont les coordonnées de x dans la base des vecteurs propres de A.
— En imposant la condition $\|x\| = 1$, nous avons :

$$\|x\|^2 = x^\top x = \left(\sum_{i=1}^{n} \alpha_i v_i\right)^\top \left(\sum_{j=1}^{n} \alpha_j v_j\right) = \sum_{i=1}^{n} \alpha_i^2 = 1.$$

— La forme quadratique $x^\top A x$ s'écrit alors en fonction des valeurs propres de A et des coefficients α_i :

$$x^\top A x = \left(\sum_{i=1}^{n} \alpha_i v_i\right)^\top A \left(\sum_{j=1}^{n} \alpha_j v_j\right) = \sum_{i=1}^{n} \alpha_i^2 \lambda_i.$$

— Puisque $\sum_{i=1}^{n} \alpha_i^2 = 1$ et que les valeurs propres λ_i sont ordonnées de manière croissante ($\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$), l'expression $\sum_{i=1}^{n} \alpha_i^2 \lambda_i$ est minimisée lorsque tout le poids est concentré sur λ_1. En d'autres termes, le minimum est atteint lorsque $x = v_1$ (c'est-à-dire lorsque $\alpha_1 = 1$ et $\alpha_i = 0$ pour $i \neq 1$).
— Dans ce cas, on a :

$$x^\top A x = \lambda_1.$$

Nous avons donc montré que :

$$\lambda_1 = \min_{\substack{x \in \mathbb{R}^n \\ \|x\|=1}} x^\top A x.$$

Réccurence :
Pour montrer que la k-ième valeur propre λ_k est donnée par :

$$\lambda_k = \min_{\substack{x \in \mathbb{R}^n \\ \|x\|=1 \\ x \perp v_1, \ldots, x \perp v_{k-1}}} x^\top A x,$$

nous allons imposer la contrainte d'orthogonalité. Il suffit de suivre le même raisonnement que pour l'initialisation en considérant que x est orthogonal aux vecteurs précédemment trouvé. ainsi ses composante selon $v_{i\in[0,k-1]}$ sont nulles et donc pour minimiser il faut $x = v_k$.

Ainsi, nous avons montré que :

$$\lambda_k = \min_{\substack{x\in\mathbb{R}^n \\ \|x\|=1 \\ x\perp v_1,\ldots,x\perp v_{k-1}}} x^\top Ax.$$

□

6.4 Démonstration de la $f^T Lf$

Preuve : Pour un signal f défini sur les nœuds d'un graphe, la forme quadratique de Laplacien $S_2(f)$ est définie comme suit :

$$S_2(f) = \frac{1}{2}\sum_{i\in V}\sum_{j\in\mathcal{N}_i} W_{i,j}\left[f(j) - f(i)\right]^2$$

Étape 1 : Simplification de la Somme Double

Dans la somme double $\sum_{i\in V}\sum_{j\in\mathcal{N}_i}$, chaque arête $(i,j) \in E$ est comptée **deux fois** : une fois dans la somme pour i avec j comme voisin, et une autre fois pour j avec i comme voisin. Pour éviter ce double comptage, nous réécrivons cette somme en comptant chaque arête une seule fois :

$$S_2(f) = \sum_{(i,j)\in E} W_{i,j}\left[f(j) - f(i)\right]^2$$

Ici, la somme est effectuée uniquement sur les arêtes (i,j) du graphe E, ce qui garantit que chaque arête est comptée une seule fois.

Étape 2 : Développement de $f^\top Lf$ en remplaçant L par $D - W$

Nous avons :

$$f^\top Lf = f^\top(D - W)f = f^\top Df - f^\top Wf$$

Développons chacun de ces termes.

Calcul du Terme $f^\top Df$

Le terme $f^\top Df$ est donné par :

$$f^\top Df = \sum_{i\in V} D_{ii} f(i)^2$$

En utilisant la définition de $D_{ii} = \sum_{j \in \mathcal{N}_i} W_{i,j}$, nous avons :

$$f^\top Df = \sum_{i \in V} \left(\sum_{j \in \mathcal{N}_i} W_{i,j} \right) f(i)^2 = \sum_{i \in V} \sum_{j \in \mathcal{N}_i} W_{i,j} f(i)^2$$

Calcul du Terme $f^\top Wf$
Le terme $f^\top Wf$ est donné par :

$$f^\top Wf = \sum_{i \in V} \sum_{j \in \mathcal{N}_i} W_{i,j} f(i) f(j)$$

Étape 3 : Assemblage de $f^\top Lf$

En substituant $f^\top Df$ et $f^\top Wf$ dans l'expression de $f^\top Lf$, nous obtenons :

$$f^\top Lf = \sum_{i \in V} \sum_{j \in \mathcal{N}_i} W_{i,j} f(i)^2 - \sum_{i \in V} \sum_{j \in \mathcal{N}_i} W_{i,j} f(i) f(j)$$

Nous pouvons maintenant regrouper les termes en fonction de $(f(i) - f(j))^2$ en utilisant l'identité :

$$f(i)^2 - f(i)f(j) + f(j)^2 - f(i)f(j) = (f(i) - f(j))^2$$

En appliquant cela à chaque terme, nous obtenons :

$$f^\top Lf = \sum_{(i,j) \in E} W_{i,j} \left[f(i)^2 - 2f(i)f(j) + f(j)^2 \right] = \sum_{(i,j) \in E} W_{i,j} \left[f(j) - f(i) \right]^2$$

ce qui est exactement la définition de $S_2(f)$. □

6.5 Démonstration Laplacien Normalisé

Preuve :détaillée du spectre du Laplacien Normalisé.

Considérons un graphe pondéré non orienté connexe $G = (V, E)$ de N sommets, avec une matrice de degré D et une matrice d'adjacence pondérée W. Le Laplacien normalisé du graphe est défini par :

$$\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}.$$

Le Laplacien normalisé $\mathcal{L}$ est une matrice symétrique, semi-définie positive, ce qui implique que toutes ses valeurs propres sont réelles et positives ou nulles.

Étape 1 : Valeurs propres de $\mathcal{L}$ et Théorème de Courant-Fischer
Pour un graphe connexe, la plus petite valeur propre de $\mathcal{L}$ est 0. On démontre cela

en utilisant la décomposition spectrale et le théorème de Courant-Fischer, qui caractérise la k-ième plus petite valeur propre d'une matrice symétrique définie positive A par :

$$\lambda_k = \min_{\substack{S \subset \mathbb{R}^N \\ \dim(S)=k}} \max_{\substack{f \in S \\ f \neq 0}} \frac{f^\top A f}{f^\top f}.$$

Appliqué à $\mathcal{L}$, cela donne :

$$\lambda_0 = \min_{\|f\|=1} f^\top \mathcal{L} f.$$

Soit $\mathbf{1}$ le vecteur constant de taille N (i.e., $\mathbf{1} = [1, 1, \ldots, 1]^\top$). On vérifie que :

$$\mathcal{L}\mathbf{1} = 0,$$

ce qui signifie que $\mathbf{1}$ est un vecteur propre associé à la valeur propre $\lambda_0 = 0$. Pour un graphe connexe (il existe au moins un chemin pour relier chaque nœuds), il est possible de démontrer que cette valeur propre est simple, c'est-à-dire qu'il n'existe qu'un seul vecteur propre associé à $\lambda_0 = 0$ (tous les vecteurs propres associés à cette valeur propre sont des multiples de $\mathbf{1}$).

Étape 2 : Borne supérieure des valeurs propres de $\mathcal{L}$

Puisque $\mathcal{L} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ et que $D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ est une matrice de similarité normalisée avec toutes ses valeurs propres dans l'intervalle $[0, 1]$(car elle est symétrique et normalisée), les valeurs propres de $\mathcal{L}$ sont dans $[0, 2]$.

- cas $\lambda_{max} = 2$: La valeur propre maximale de $\mathcal{L}$ atteint 2 si et seulement si le graphe est biparti. Dans ce cas, les sommets peuvent être divisés en deux sous-ensembles disjoints V_1 et V_2 de sorte que toutes les arêtes connectent un sommet de V_1 à un sommet de V_2. Cela signifie que le vecteur qui alterne les signes entre les sommets des deux sous-ensembles est un vecteur propre associé à la valeur propre $\lambda_{\max} = 2$.

Conclusion :

Ainsi, pour un graphe connexe, les valeurs propres du Laplacien normalisé $\mathcal{L}$ sont comprises entre 0 et 2, avec :

$$0 = \lambda_0 < \lambda_1 \leq \cdots \leq \lambda_{\max} \leq 2.$$

La valeur $\lambda_{\max} = 2$ est atteinte si et seulement si le graphe est biparti, auquel cas le vecteur propre correspondant alterne les signes entre les deux sous-ensembles de sommets.

6.6 Calcul de la bandwith

Exemple : Exemple : Bande passante pour des signaux à une ou plusieurs composantes.
Nous allons considérer deux cas pour illustrer les situations de petite et grande bande passante :

- Considérons que $G(n,k)$ matérialise l'énergie d'un signal multi-composante.
- Nous étudierons un signal à **une composante** (petite bande passante) et un signal à **deux composantes** (grande bande passante).

Cas 1 : Signal à une composante. Supposons que $G(n,k)$ est concentrée uniquement autour d'une valeur propre λ_{k_0}. Cela signifie que :

$$G(n,k) = \begin{cases} A, & \text{si } k = k_0, \\ 0, & \text{sinon.} \end{cases}$$

Ici, toute l'énergie est concentrée autour de λ_{k_0}. Calculons la bande passante dans ce cas.

- La smoothness locale $\lambda(n)$ est donnée par :

$$\lambda(n) = \frac{\sum_{k=1}^{N} \lambda_k G(n,k)}{\sum_{k=1}^{N} G(n,k)} = \frac{\lambda_{k_0} A}{A} = \lambda_{k_0}.$$

- La bande passante est donnée par :

$$\text{Bande passante} = \sqrt{\frac{\sum_{k=1}^{N} (\lambda_k - \lambda(n))^2 G(n,k)}{\sum_{k=1}^{N} G(n,k)}}.$$

  Comme $G(n,k)$ est nulle pour $k \neq k_0$, cela donne :

$$\text{Bande passante} = \sqrt{\frac{(\lambda_{k_0} - \lambda_{k_0})^2 A}{A}} = 0.$$

Ce résultat montre que la bande passante est nulle, car toute l'énergie est concentrée sur une seule valeur propre. Physiquement, cela indique un signal parfaitement lisse, sans aucune variation spectrale.

Exemple :

Cas 2 : Signal à deux composantes. Supposons maintenant que $G(n,k)$ est répartie sur deux valeurs propres λ_{k_1} et λ_{k_2}. Cela signifie que :

$$G(n,k) = \begin{cases} A_1, & \text{si } k = k_1, \\ A_2, & \text{si } k = k_2, \\ 0, & \text{sinon.} \end{cases}$$

Ici, l'énergie est répartie entre deux composantes spectrales. Calculons la bande passante dans ce cas.

— La smoothness locale $\lambda(n)$ est donnée par :

$$\lambda(n) = \frac{\sum_{k=1}^{N} \lambda_k G(n,k)}{\sum_{k=1}^{N} G(n,k)} = \frac{\lambda_{k_1} A_1 + \lambda_{k_2} A_2}{A_1 + A_2}.$$

— La bande passante est donnée par :

$$\text{Bande passante} = \sqrt{\frac{\sum_{k=1}^{N} (\lambda_k - \lambda(n))^2 G(n,k)}{\sum_{k=1}^{N} G(n,k)}}.$$

Substituons $G(n,k)$:

$$\text{Bande passante} = \sqrt{\frac{(\lambda_{k_1} - \lambda(n))^2 A_1 + (\lambda_{k_2} - \lambda(n))^2 A_2}{A_1 + A_2}}.$$

Ce résultat montre que la bande passante est proportionnelle à l'écart quadratique moyen entre les deux valeurs propres λ_{k_1} et λ_{k_2}, pondéré par leurs énergies respectives A_1 et A_2. Physiquement :

— Une large bande passante indique que le signal contient des contributions spectrales significatives à plusieurs échelles (λ_{k_1} et λ_{k_2}).

— Cela traduit une plus grande complexité du signal, avec des variations plus rapides ou locales.

6.7 Description des test de l'algorithme Scalar Analyzer

— **Tests associés au DataLoader :**
  — **Test 1 : Téléchargement des données**
    — Vérifier que les données sont correctement téléchargées.
    — Afficher les données sur plusieurs instants de temps, nœuds et types de données.
— **Tests associés au Graph Builder :**
  — **Test 1 : Génération de graphe**

  - Générer un graphe en anneaux et afficher sa forme.
  - Vérifier l'affichage de la matrice d'adjacence.
- **Test 2 : Normalisation**
  - Vérifier les coefficients de la matrice d'adjacence W.
  - Vérifier le Laplacien normalisé dans le cas d'un graphe en anneaux.
- **Test 3 : Filtrage du graphe**
  - Vérifier les algorithmes de filtrage de graphes.
  - Utiliser des matrices d'adjacence générées aléatoirement.

— **Tests associés au Scalar Analyzer :**

- **Test 1 : Signal**
  - Vérifier que le signal est bien centré et réduit.
- **Test 2 : Smoothness (Lissage)**
  - Comparer le calcul de la smoothness matricielle et de la somme des poids.
  - Calculer la smoothness sur un graphe en anneaux pour un signal affine.
- **Test 3 : GFT (Graph Fourier Transform)**
  - Calculer la transformée de Fourier sur un graphe en anneaux.
  - Afficher les trois premiers vecteurs propres.
- **Test 4 : Local Smoothness (Lissage local)**
  - Vérifier la local smoothness pour un vecteur propre du graphe et un signal affine.
- **Test 5 : Bandwidth (Bande passante)**
  - Vérifier la bande passante pour un signal égal aux valeurs propres du Laplacien.

6.8 Démonstration de la valeurs de la moyenne pour un bruit blanc gaussien connaissant le graphe

Preuve : Nous cherchons à démontrer que, pour un bruit blanc gaussien f centré et réduit sur un graphe ayant n nœuds, l'espérance de la *smoothness* est donnée par :

$$\mathbb{E}[S(f) \mid W] = 1 + \frac{1}{n}.$$

Étape 1 : Expression de la smoothness La *smoothness* d'un signal f sur un graphe pondéré est définie par :

$$S(f) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j}(f_i - f_j)^2,$$

où $w_{i,j}$ est le poids de l'arête reliant les nœuds i et j, et f_i est la valeur du signal au nœud i.

Pour un signal normalisé, nous avons :

$$f_i^{\text{normalized}} = \frac{f_i - \bar{f}}{\sigma_f},$$

avec :

$$\bar{f} = \frac{1}{n}\sum_{k=1}^{n} f_k \quad \text{et} \quad \sigma_f^2 = \frac{1}{n}\sum_{k=1}^{n}(f_k - \bar{f})^2.$$

Étape 2 : Remplacement des f_i normalisés dans $S(f)$ En substituant les valeurs normalisées $f_i^{\text{normalized}}$ dans l'expression de la *smoothness*, on obtient :

$$S(f) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j}\left(\frac{f_i - \bar{f}}{\sigma_f} - \frac{f_j - \bar{f}}{\sigma_f}\right)^2.$$

Simplifions cette expression :

$$S(f) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j}\frac{(f_i - f_j)^2}{\sigma_f^2}.$$

Étape 3 : Espérance de $S(f)$ pour un bruit blanc gaussien Pour un bruit blanc gaussien unitaire, $f_i \sim \mathcal{N}(0,1)$, et les f_i sont i.i.d. Cela implique :

$$\mathbb{E}[f_i] = 0, \quad \mathbb{E}[f_i^2] = 1, \quad \text{et} \quad \mathbb{E}[f_i f_j] = 0 \,\text{pour } i \neq j.$$

L'espérance conditionnelle de $S(f)$ devient alors :

$$\mathbb{E}[S(f) \mid W] = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j}\mathbb{E}\left[\frac{(f_i - f_j)^2}{\sigma_f^2}\right].$$

Développons $(f_i - f_j)^2$:

$$(f_i - f_j)^2 = f_i^2 + f_j^2 - 2f_i f_j.$$

En prenant l'espérance :

$$\mathbb{E}[(f_i - f_j)^2] = \mathbb{E}[f_i^2] + \mathbb{E}[f_j^2] - 2\mathbb{E}[f_i f_j] = 1 + 1 - 0 = 2.$$

Ainsi, l'espérance devient :

$$\mathbb{E}[S(f) \mid W] = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j} \cdot \frac{2}{\mathbb{E}[\sigma_f^2]}.$$

Étape 4 : Espérance de σ_f^2 La variance σ_f^2 est donnée par :

$$\sigma_f^2 = \frac{1}{n}\sum_{k=1}^{n} f_k^2 - \bar{f}^2,$$

où :

$$\bar{f}^2 = \left(\frac{1}{n} \sum_{k=1}^{n} f_k \right)^2 .$$

Pour un bruit blanc gaussien unitaire, $\mathbb{E}[f_k^2] = 1$ et $\mathbb{E}[\bar{f}^2] = \frac{1}{n}$. Ainsi, l'espérance de σ_f^2 est donnée par :

$$\mathbb{E}[\sigma_f^2] = \frac{1}{n} \sum_{k=1}^{n} \mathbb{E}[f_k^2] - \mathbb{E}[\bar{f}^2] = 1 - \frac{1}{n}.$$

Étape 5 : Substitution de $\mathbb{E}[\sigma_f^2]$ dans $S(f)$ En remplaçant $\mathbb{E}[\sigma_f^2]$ dans l'expression de $\mathbb{E}[S(f) \mid W]$, nous obtenons :

$$\mathbb{E}[S(f) \mid W] = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j} \cdot \frac{2}{1 - \frac{1}{n}}.$$

Puisque $\sum_{i=1}^{n} \sum_{j=1}^{n} w_{i,j} = 1$, cela donne :

$$\mathbb{E}[S(f) \mid W] = \frac{2}{1 - \frac{1}{n}} = 1 + \frac{1}{n}.$$

Conclusion : L'espérance de la *smoothness* pour un bruit blanc gaussien unitaire, centré et réduit, est donnée par :

$$\mathbb{E}[S(f) \mid W] = 1 + \frac{1}{n}.$$

□