



École Nationale Supérieure d’Informatique pour l’Industrie et l’Entreprise

Trend Filtering sur Graphe

Élève : Dorian Joubaud
Tuteur académique à l’ENSIIE : Massimina Merabet



Entreprise : École Normale Supérieure - Centre Borelli
Maitre de stage : Charles Truong

Stage du mardi 01 juin 2021 au vendredi 13 août 2021

Remerciements

Je tiens à remercier M. Charles TRUONG, chercheur au centre Borelli, pour m'avoir aidé et conseillé durant ce stage.

Je tiens également à remercier Mme Mathilde Mougeot, enseignante à l'ENSIIE, affiliée à l'ENS Paris-Saclay, pour avoir eu confiance en moi pour ce stage et M. Simon Lavergne, ami et collègue de promo à l'ENSIIE, pour avoir passé ce stage avec moi.

Ce fut un véritable plaisir de travailler dans la bonne humeur avec eux.

ENS Paris-Saclay - Centre Borelli

Le Centre Borelli est un laboratoire de recherche interdisciplinaire qui explore différentes thématiques de recherche, autour de la modélisation/simulation physique et numérique (fluides complexes, dynamiques moléculaires), de l'étude de la sensorimotricité, de la perception artificielle (images et signaux) et de l'apprentissage.

Le Centre Borelli est placé sous la tutelle du Centre Nationale de la Recherche Scientifique (CNRS), de l'École Nationale Supérieure Paris-Saclay, du Service de Santé des Armées (SSA), de l'université de Paris et de l'Inserm.

Il est issu de la fusion de deux unités CNRS, le CMLA et Cognac G réunit des chercheurs en mathématiques, en informatique et en neurosciences. Il incarne une collaboration inédite entre les instituts INSMI (Institut national des sciences mathématiques et de leurs interactions) et INSB (Institut des sciences biologiques).



Sommaire

1	Introduction	1
2	Trend Filtering sur Graphe	2
2.1	Trend filtering univarié	2
2.2	Trend filtering sur graphe	2
2.3	Apprentissage par morceaux polynômial	3
3	Implémentations	5
3.1	CVXOPT	5
3.2	Création de graphes pour le Trend Filtering	6
3.2.1	Création de graphes sous Python	6
3.2.2	Signal sur graphes	6
3.3	Résolution de cas "simple"	7
3.4	Visualisation des ruptures	8
3.4.1	k pair	8
3.4.2	k impair	8
3.4.3	Exemple	8
3.5	Optimisation du code	8
4	Paramétrisation	9
4.1	Poids de la pénalité λ	9
4.1.1	λ_{max}	9
4.2	Précision	11
5	Application	12
5.1	Le data set	12
5.1.1	Présentation	12
5.1.2	Mise en forme des données	12
5.1.3	Création du Graphe	14
5.2	Étude sur les signaux	15
5.2.1	Apprentissage constant par morceaux	16
5.3	Calibration	17
5.3.1	λ	17
5.3.2	Regroupement en clusters	17
5.4	Smoothness	19
5.5	Interpolation de noeuds	20
5.5.1	Approche par moyenne empirique	20
5.5.2	Approche par moyenne des prédictions	20
5.6	Simplification du graphe	21
6	Conclusion	23
A	Annexe : Développement durable et responsabilité sociétale	24
A.1	Développement durable	24

1 Introduction

Le service national Français de météorologie (météo France) a rendu publique un data set de plusieurs données météorologiques (température, vitesse et direction du vent, humidité, etc..) mesurées toutes les heures en Bretagne, durant le mois de janvier 2014. Les contrastes importants des données météorologiques qui peuvent être observés entre l'Ouest et l'Est, le Nord et le Sud et entre le littoral et l'intérieur témoigne de la richesse et de la diversité du climat qui règne sur ce territoire.

Dès lors, en considérant chaque station météorologique comme un noeud d'un graphe et les données mesurées par celles ci comme des signaux les parcourant, il est pertinent de travailler avec ces données dans le cadre de l'apprentissage et la modélisation mathématique d'un signal parcourant un graphe.

Le trend filtering est une branche du traitement de signal. Proposé par Ryan J. Tibshirani en 2014 pour la régression non paramétrique, il vise à approcher n'importe quelle série temporelle avec une représentation polynomiale par morceaux.

L'objectif de ce stage est d'implémenter et d'optimiser un modèle de trend filtering qui sera étendu à la théorie des graphes. De sorte que l'on puisse utiliser ce modèle sur le graphe de la Bretagne afin de découvrir quantitativement les structures spatiales de cette région. On s'intéressera particulièrement à la variable de température et par conséquent à un traitement univarié du problème.

Dans un second temps, on tentera d'utiliser le modèle dans le but de simplifier au maximum le graphe sans perte d'information. Pour ce faire, on essayera d'interpoler les noeuds du graphe.

Les connaissances mathématiques nécessaires à la réalisation de ce modèle étant très spécifiques et avancés, nous nous appuierons sur plusieurs articles, cités dans la bibliographie, traitant de ces sujets.

Les fichiers code du stage ont été développés sur des Jupyter Notebook et sont présent sur le Git [DorianJoubaud](#) ;

2 Trend Filtering sur Graphe

Dans un premier temps, nous allons définir la base théorique des objets mathématiques utilisés durant ce stage et les présenter à l'aide d'exemples simples. Aussi, nous aborderons les différents paramètres inhérents à nos modèles, comment les définir et comment bien les calibrer. Dans un second temps, nous verrons l'application de nos modèles sur les données météorologiques de la Bretagne.

Définissons tout d'abord formellement le trend filtering.

On s'inspirera, pour cela, du papier de recherche **Trend Filtering on graphs** du *Journal of Machine Learning Research*. Cet article traite, notamment, des estimateurs adaptatifs basés sur la pénalisation ℓ_1 pour la différence discrète sur des graphes. Ceci généralise l'idée du trend filtering "classique" pour la régression non paramétrique aux graphes.

Pour ce faire, on définit un problème d'optimisation convexe qui peut être facilement résolu.

2.1 Trend filtering univarié

Soient y un signal tel que $y = (y_1, \dots, y_n) \in \mathbb{R}^n$ et $k > 0$, on définit **l'apprentissage par trend filtering d'ordre k** : $\hat{\beta} = (\hat{\beta}_1, \dots, \hat{\beta}_n)$ par

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2} \|y - \beta\|_2^2 + \lambda \|D^{(k+1)}\beta\|_1$$

avec $\lambda \geq 0$ l'hyperparamètre de régularisation, et $D^{(k+1)}$ l'opérateur de différence finie d'ordre $k + 1$.

Pour $k = 0$,

$$D^{(1)} = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \vdots & & \ddots & \ddots & \\ 0 & 0 & \dots & -1 & 1 \end{bmatrix}$$

On définit alors récursivement $D^{(k+1)} = D^{(k)}D^{(1)}$

2.2 Trend filtering sur graphe

Soient $G = (V, E)$ un graphe, de sommets $V = \{1, \dots, n\}$, d'arêtes non-orientées $E = \{e_1, \dots, e_m\}$ et $y = (y_1, \dots, y_n) \in \mathbb{R}^n$ parcourant le graphe. On définit **l'apprentissage par trend filtering d'ordre k sur graphe** : $\hat{\beta} = (\hat{\beta}_1, \dots, \hat{\beta}_n)$ par

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2} \|y - \beta\|_2^2 + \lambda \|\Delta^{(k+1)}\beta\|_1$$

Pour $k = 0$, on définit l'opérateur de différence finie d'ordre k $\Delta^{(1)}$ de telle sorte qu'elle donne l'équivalent graphique d'une pénalité sur les différences locales :

$$\|\Delta^{(1)}\beta\|_1 = \sum_{(i,j) \in E} |\beta_i - \beta_j|$$

Soit $\Delta_l^{(1)} \in \{-1, 0, 1\}^{m*n}$ la matrice d'adjacence du graphe (orienté) G , contenant une ligne pour chaque arête du graphe. ex : si $e_l = (i, j)$, alors $\Delta^{(1)}$ à pour l ième ligne

$$\Delta_l^{(1)} = (0, \dots, \underset{i}{\overset{\uparrow}{-1}}, \dots, \underset{j}{\overset{\uparrow}{1}}, \dots, 0)$$

où l'orientation du graphe est arbitraire.

Pour $k \geq 1$, on définit récursivement les ordres supérieurs de Δ (de la même manière que dans le cas univarié). La récursion alterne en multipliant par le premier opérateur de différence $\Delta^{(1)}$ et sa transposée (en tenant compte que cette matrice n'est pas carrée).

$$\Delta^{(k+1)} = \begin{cases} (\Delta^{(1)})^T \Delta^{(k)} & \text{pour } k \text{ impair} \\ \Delta^{(1)} \Delta^{(k)} & \text{pour } k \text{ pair} \end{cases}$$

On remarquera que $\Delta^{(k+1)} \in \mathbb{R}^{n*n}$ pour k impair et que $\Delta^{(k+1)} \in \mathbb{R}^{m*n}$ pour k pair.

2.3 Apprentissage par morceaux polynômial

L'ordre $k + 1$ de l'opérateur Δ défini ci-dessus est, par construction, lié aux degrés des polynômes qui définissent le signal appris (qui est polynômial par morceaux). On remarque notamment que la présence de composante nulle dans $\Delta^{(k+1)}\beta$ implique un ordre k spécifique dans la structure polynômiale par morceau du signal, plus précisément :

- **Constant par morceau** ($k = 0$) : On dit qu'un signal β est constant par morceaux sur un graphe G si un grand nombre des $\beta_i - \beta_j$ sont nulles sur les arêtes $(i, j) \in E$ dans G . Ce qui revient à la présence de composante nulle dans $\Delta^{(1)}\beta$. (On remarquera que $\Delta^{(1)} = D$, la matrice d'adjacence de G).
- **Linéaire par morceau** ($k = 1$) : On dit qu'un signal β a une structure linéaire par morceaux sur G si β satisfait

$$\beta_i - \frac{1}{n_i} \sum_{(i,j) \in E} \beta_j = 0,$$

pour un grand nombre de noeud $i \in V$, où n_i est le nombre de noeuds connectés à i . De même, cela revient à la présence de composante nulle dans $\Delta^{(2)}\beta$. (On remarquera que $\Delta^{(2)} = L$, le Laplacien de G).

- **Polynomial d'ordre supérieur par morceau** ($k \geq 2$) : On dit qu'un signal β a une structure quadratique par morceau sur G , si les $\alpha_i - \alpha_j$ (différence d'ordre 1), où $\alpha = \Delta^{(2)}\beta$ (différence d'ordre 2), sont majoritairement nulles sur les arêtes $(i, j) \in E$. De même, β a une structure cubique par morceau sur G , si les $\alpha_i - \frac{1}{n_i} \sum_{(i,j) \in E} \alpha_j$

(différence d'ordre 2), où $\alpha = \Delta^{(2)}\beta$ (différence d'ordre 2), sont majoritairement nulles sur les noeuds $i \in V$.

On étend alors cette propriété, alternant entre différences d'ordre 1 et 2, selon la parité de k .

3 Implémentations

Le langage de programmation Python offre une grande palette d'outils permettant la résolution de problèmes d'optimisation complexes. **CVXOPT** est une librairie d'optimisation convexe basée sous Python. On se propose d'implémenter la résolution de notre problème dit "des moindres carrés" régularisée selon la norme ℓ_1 .

Le solver transforme le problème initial (\mathcal{P}) en un problème dual (\mathcal{D}) de forme quadratique (QP) et le résout à l'aide de la méthode des points intérieurs.

$$(\mathcal{P}) \left\{ \begin{array}{l} \min \|Ax - b\|_2^2 + \|x\|_1 \\ x \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m \end{array} \right. \Leftrightarrow (\mathcal{D}) \left\{ \begin{array}{l} \min \|Ax - b\|_2^2 + 1^T v \\ x \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m \\ \text{s.c. } -v \preceq x \preceq v \end{array} \right.$$

3.1 CVXOPT

Dans le but de correctement appréhender le Trend Filtering sur graphes, utilisons CVXOPT afin de résoudre une problématique de Trend Filtering classique.

On pose :

$$(\mathcal{P}_c) \quad \operatorname{argmin}_{x \in \mathbb{R}^n} \frac{1}{2} \|y - x\|_2^2 + \lambda \|Dx\|_1$$

$$\text{avec } y \in \mathbb{R}^n, \lambda \geq 0 \text{ et } D = \begin{bmatrix} 0 & \dots & 0 & -1 & 2 & -1 & 0 & \dots & 0 \end{bmatrix}$$

On cherche donc ici à approximer le signal y par un signal x polynomial par morceau. On prendra y un signal exemple fourni par CVXOPT et $\lambda = 50$.

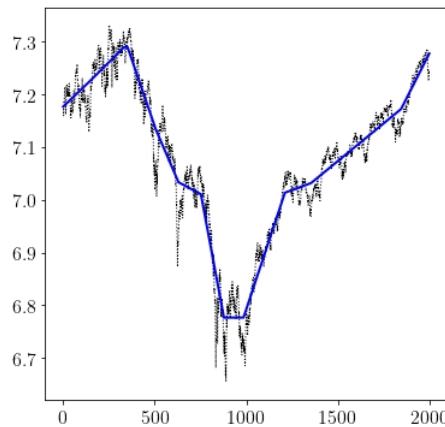


FIGURE 1 – Représentation du trend filtering d'ordre linéaire ($k = 1$)

On approche ici la série temporelle (y) en noir par la courbe (x), linéaire par morceaux, en bleu.

3.2 Création de graphes pour le Trend Filtering

3.2.1 Création de graphes sous Python

Sous python, plusieurs librairie permettent la génération de graphes (géométriques, erdos-renyi, etc ...), nous utiliserons **PYGSP** et **Networkx**.

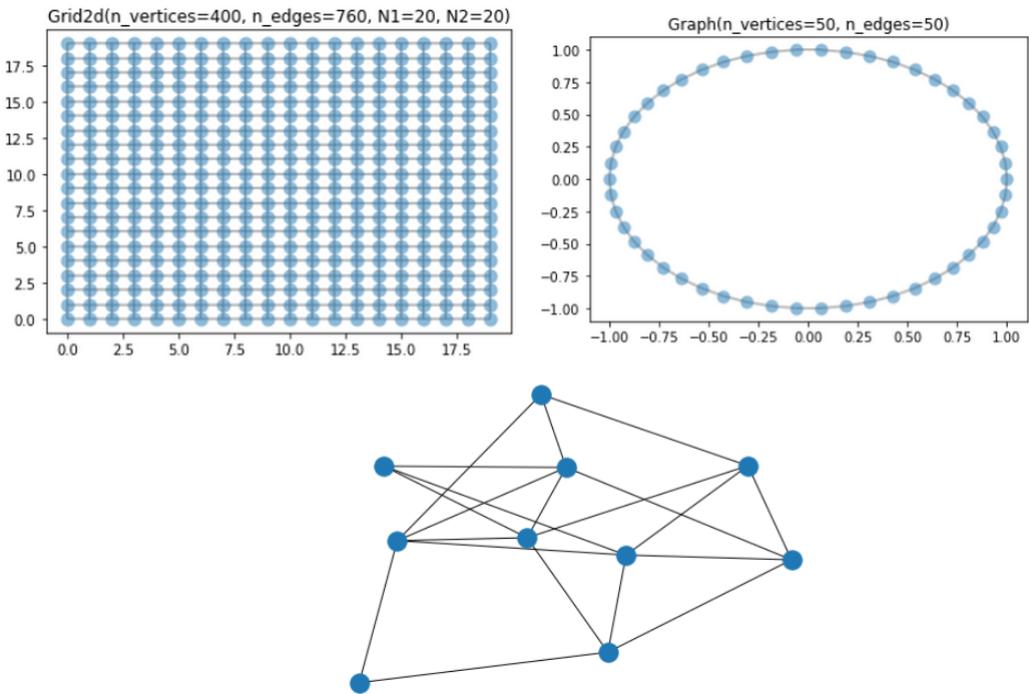


FIGURE 2 – Exemples de graphes

3.2.2 Signal sur graphes

Considérons un signal de longueur n , égale au nombre de noeuds dans un graphe G . En assignant à chaque noeuds une valeur, on crée un signal parcourant le graphe G . On représente ici, le signal $s(x) = 5 * \sin(x)$ sur $[0; 6\pi]$ (pour 50 noeuds en cercle et pour 400 noeuds en grille)

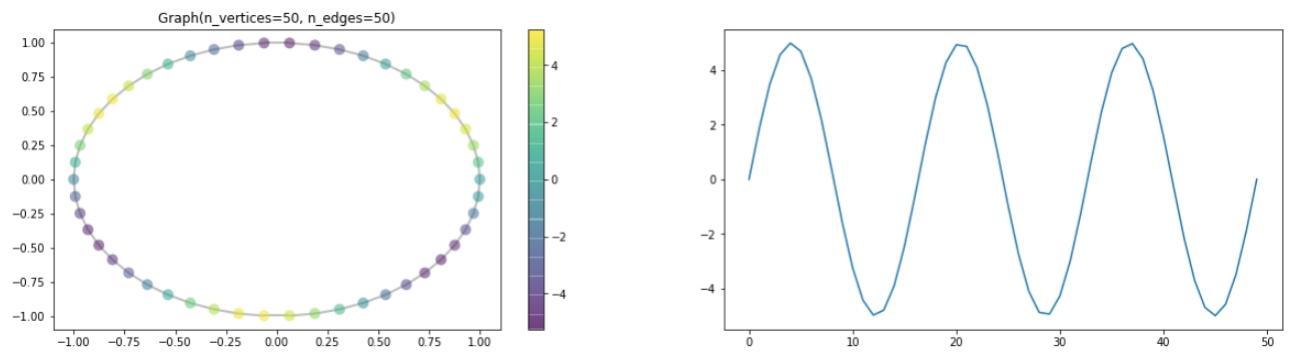


FIGURE 3 – Signal sur graphe circulaire

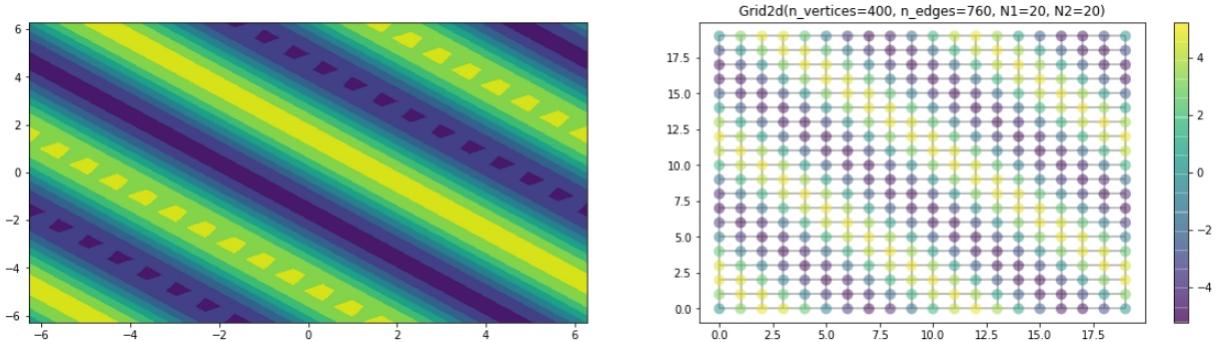


FIGURE 4 – Signal sur graphe "grille"

3.3 Résolution de cas "simple"

Considérons un graphe grille 20×20 et un signal $z = x + y$ (non bruité) le parcourant (avec $(x, y) \in [0; 100]^2$)

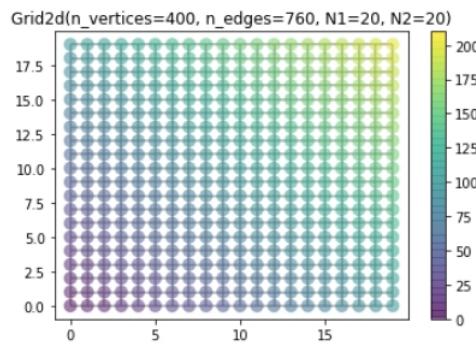


FIGURE 5 – Signal linéaire sur graphe grille

Notre objectif est donc d'approcher ce signal sur graphe par morceaux polynomiaux. On obtient le signal approché x représenté ci dessous.

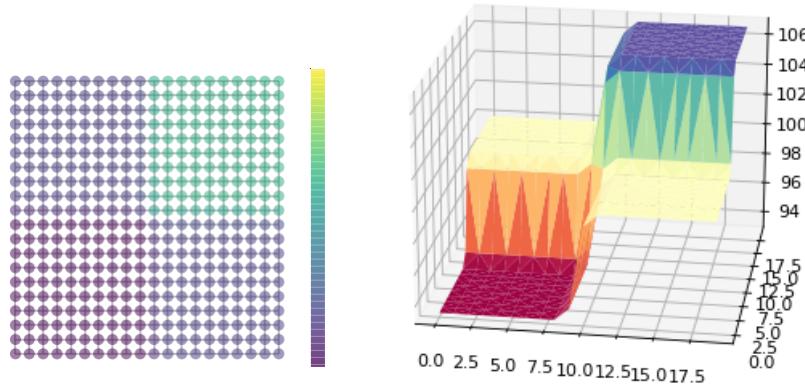


FIGURE 6 – Signal approché constant par morceaux

3.4 Visualisation des ruptures

Le signal approché étant composé de plusieurs polynômes (de mêmes degré), la présence de composante nulle dans $\Delta^{(k+1)}\beta$ (telle que définit dans la partie **2.3 Apprentissage par morceaux polynomial**), nous renseigne sur les frontières entre ces polynômes. En effet, une composante (ou différence) non-nulle définit un changement, non négligeable, d'allure du signal et donc de polynôme. On souhaite alors représenter ces ruptures sur les graphes. Pour ce faire, on remarque que $\Delta^{(k+1)} \in \mathbb{R}^{n*n}$ pour k impair et que $\Delta^{(k+1)} \in \mathbb{R}^{m*n}$ pour k pair..

3.4.1 k pair

Dans le cas où k est pair, $\Delta^{(k+1)}\beta \in \mathbb{R}^m$, avec m correspondant au nombre d'arêtes du graphe G . On visualisera les arêtes de différence non-nulle en rouge.

3.4.2 k impair

Dans le cas où k est pair, $\Delta^{(k+1)}\beta \in \mathbb{R}^n$, avec n correspondant au nombre de noeuds du graphe G . On visualisera les arêtes adjacentes aux noeuds de différence non-nulle en rouge.

3.4.3 Exemple

Considérons le signal $\mathcal{D} = x^2 + y^2$ parcourant le graphe grille (de taille 10x10) G . On souhaite approcher \mathcal{D} par \mathcal{J} un signal constant par morceaux (donc $k = 0$).

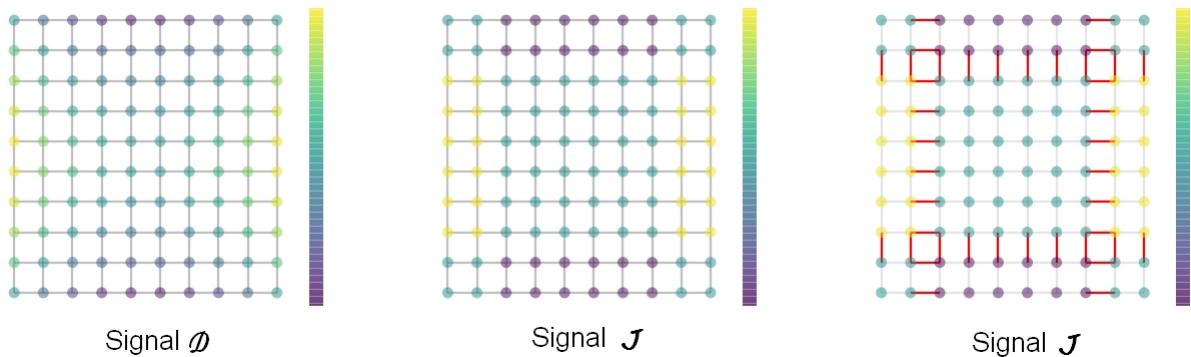


FIGURE 7 – Visualisation des ruptures

3.5 Optimisation du code

Dans l'objectif de gagner en clarté et en temps de calcul, nous utilisons le langage Python orienté objet. La classe **TFOG** d'attributs k (ordre de l'apprentissage), $signal$, $graph$ et $graph_size$, contient les méthodes permettant la résolution de problème de Trend Filtering sur graphe ainsi que l'affichage des résultats obtenus. De plus, la classe *Graph* hérite de la classe *Graphs* du module PYGSP, cette nouvelle classe permet la création de graphes aléatoires et de graphes grilles. Le diagramme des classes correspondant se trouve en annexe (Figure 29).

4 Paramétrisation

Plusieurs paramètres intrinsèques à notre problème sont importants. En effet, bien comprendre la nature de ces paramètres permet de mieux appréhender le problème initial et ainsi mieux le résoudre.

4.1 Poids de la pénalité λ

On rappel :

Soient $G = (V, E)$ un graphe, de sommets $V = \{1, \dots, n\}$, d'arêtes non-orientées $E = \{e_1, \dots, e_m\}$ et $y = (y_1, \dots, y_n) \in \mathbb{R}^n$ parcourant le graphe. On définit l'**apprentissage par trend filtering d'ordre k sur graphe** : $\hat{\beta} = (\hat{\beta}_1, \dots, \hat{\beta}_n)$ par

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2} \|y - \beta\|_2^2 + \lambda \|\Delta^{(k+1)}\beta\|_1$$

Le paramètre λ représente le poids de la pénalité ℓ_1 .

Suite à l'ajout des arêtes rouges, on remarque que λ a un impact direct sur l'apparition de celles-ci et donc sur la présence de composantes non-nulles de $\Delta^{(k+1)}\beta$. En effet, le nombre de composantes non-nulles est plus faible quand λ augmente. Finalement on notera λ_{max} la plus petite valeur telle que $\Delta^{(k+1)}\beta$ vaut le vecteur nul. Dès lors, on prendra λ dans $[0; \lambda_{max}]$, les valeurs $\geq \lambda_{max}$ ne comportant aucune informations supplémentaires pour notre problème.

Prenons comme exemple, le signal $\mathcal{S} = x^2 - y$ sur $[-10; 10]$ sur un graphe grille 10×10 G . Après détermination de $\lambda_{max} \approx 79$ (voir 4.1.1), on détermine le signal appris avec des valeurs inférieures à λ_{max} .

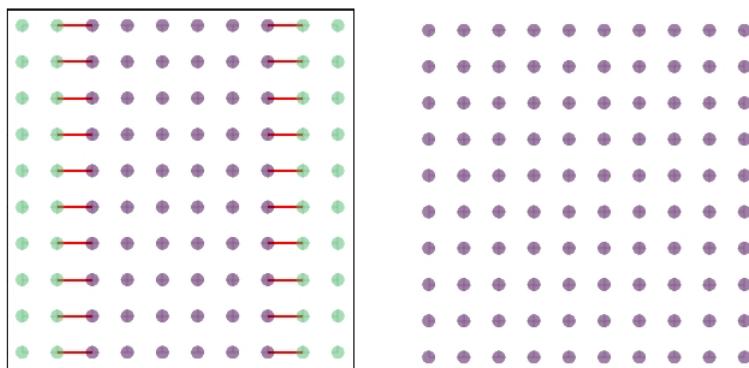


FIGURE 8 – Signal appris pour $\lambda = 78$ et $\lambda \geq \lambda_{max}$

Les noeuds de même couleurs sont de mêmes valeurs.

4.1.1 λ_{max}

Afin de réduire notre champs d'étude à $[0; \lambda_{max}]$, il est nécessaire de pouvoir calculer la valeur de λ_{max} .

Théorème 4.1.1

Soit (\mathcal{P}) le problème d'optimisation défini en 2.2

$$(\mathcal{P}) \left\{ \begin{array}{l} \hat{\beta} = \underset{\beta \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2} \|y - \beta\|_2^2 + \lambda \|\Delta^{(k+1)} \beta\|_1 \end{array} \right.$$

Il existe $\lambda_{max} \in \mathbb{R}$ tel que, pour tout $\lambda \geq \lambda_{max}$, $\Delta^{(k+1)} \hat{\beta} = 0$ et $\lambda_{max} = \|\Delta^{(k+1)-T} y\|_\infty$

Preuve

Posons $(\mathcal{D}) \left\{ \begin{array}{l} \hat{\delta} = \underset{\delta \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2} \|y - \nabla \delta\|_2^2 + \lambda \|\delta\|_1 \end{array} \right.$

et $f(\delta) = \frac{1}{2} \|y - \nabla \delta\|_2^2 + \lambda \|\delta\|_1$, f est convexe, par condition du premier ordre sans contrainte, on a :

$$\frac{\partial f}{\partial \delta}(\hat{\delta}) = 0$$

On obtient alors

$$-\nabla^T(y - \nabla \hat{\delta}) = \lambda \hat{z}_\lambda$$

avec \hat{z}_λ la variable duale tel que $\hat{z}_{\lambda,j} = \begin{cases} \operatorname{sgn}(\hat{\delta}_j) & \text{si } \hat{\delta}_j \neq 0 \\ x \in [-1; 1] & \text{si } \hat{\delta}_j = 0 \end{cases}$

Or tant que $\hat{\delta} = 0$, on a :

$$-\nabla^T y = \lambda \hat{z}_\lambda$$

On obtient donc :

$$\|\nabla^T y\|_\infty = \lambda \|\hat{z}_\lambda\|_\infty$$

Si $\|\hat{z}_\lambda\|_\infty \neq 1$, alors λ pourrait diminuer, avec $\|\hat{z}_\lambda\|_\infty$ qui augmenterait pour maintenir l'égalité donnée par $\hat{\delta} = 0$.

Par conséquent, il existe λ_{max} , la plus petite valeur de λ qui maintient $\hat{\delta} = 0$ et quand $\lambda = \lambda_{max}$ on a :

$$\|\nabla^T y\|_\infty = \lambda \cdot 1 = \lambda = \lambda_{max}$$

Finalement, en posant $\delta = \Delta^{(k+1)} \beta$ et $\nabla = \Delta^{(k+1)-1}$ on revient au problème (\mathcal{P}) et on obtient :

$$\lambda_{max} = \|\Delta^{(k+1)-T} y\|_\infty$$

□

L'hypothèse $\hat{\delta} = \Delta^{(k+1)} \hat{\beta} = 0$ correspond à l'absence de rupture dans notre modèle. Ainsi tant qu'aucune rupture n'est présente, on diminue λ tout en maintenant l'égalité $\|\nabla^T y\|_\infty = \lambda \|\hat{z}_\lambda\|_\infty$.

On remarque λ_{max} est indépendant de $\hat{\beta}$, on peut donc le calculer en amont de la résolution du problème.

À titre d'exemple, pour le signal \mathcal{S} (voir 4.1), on trouve $\lambda_{max} = 79.0123456790127$ (avec $k = 0$).

4.2 Précision

La détermination d'élément nul dans $\Delta^{(k+1)}\beta$ nécessite l'introduction d'une variable *précision*. En effet, les différents calculs de nombre à virgule (encodés par des *flottant*) amènent des erreurs. Ainsi, on trouvera des composantes censées être nulles, égales à des valeurs proches de 0 (1e-7 par exemple). De plus, si le signal d'entrée comporte des valeurs assez faibles, il faut pouvoir différencier les composantes censées être nulles et celles de faible valeur.

Dans le cas de notre étude on fixera *précision* à $1e^{-4}$.

5 Application

La partie théorique du trend filtering sur graphe ainsi que son implémentation étant définies, utilisons les sur le data set des données météorologiques de la Bretagne.

5.1 Le data set

5.1.1 Présentation

Le **data set** est fourni par Météo France. Il présente les relevés de température, de d'humidité, de vitesse du vent, etc. Mais aussi les coordonnées GPS des stations météorologiques utilisées. Le data set détaillé, avec les noms de variables, types et unités est présent en annexe 30.

5.1.2 Mise en forme des données

Nous utilisons le module **Pandas** et **Geopandas** pour traiter nos données. **Geopandas**, qui hérite de **Pandas**, est très utile pour travailler avec des données géospatiales.

Notre étude ne traitant que le cas univarié, nous allons séparer le data set en tableaux de données pour chaque variable.

Pour ce faire, nous créons la variable **stations_gdf**(GeoDataFrame avec Geopandas) répertoriant le Nom, la latitude, longitude, etc.

	Numéro	Nom	Latitude	Longitude	X (Lambert II étendu)	Y (Lambert II étendu)	Altitude	geometry
0	22016001	ILE-DE-BREHAT	48.855167	-3.004500	208083	2441852	25	POINT (-3.00450 48.855167)
1	22092001	KERPERT	48.404000	-3.147667	194096	2392507	281	POINT (-3.14767 48.404000)
2	22113006	LANNAERO	48.755333	-3.468667	173267	2433190	85	POINT (-3.46867 48.75533)
3	22135001	LOUARGAT	48.551667	-3.376833	178359	2410097	148	POINT (-3.37683 48.551667)
4	22147006	MERDRIGNAC	48.182667	-2.410833	247067	2364385	131	POINT (-2.41083 48.182667)

FIGURE 9 – stations_gdf

Ainsi pour chaque variable que nous souhaitons étudier, nous créons une variable (DataFrame avec Pandas) **variable_df**, indexant la valeur associée à chaque station selon la date.

Par conséquent, si nous voulons étudier la variable *var* à la *n*ième heure, sur toutes les stations, il nous suffit d'appeler **var_df[n]**.

station_name	ARZAL	AURAY	BELLE ILE LE TALUT	BIGNAN	BREST- GUIPAVAS	BRIGNOGAN	DINARD	GUERANDE	ILE DE GROIX	ILE DE BREHAT	—	SAINTE- CAST LE G	SARZEAU SA	SIBIRI SA	SIZUN	SPEZET	ST BRIEUC	ST NAZAIRE- MONTOIR	ST- SEGALS THEIX	VANNES SENE	
date																					
2014-01-01 00:00:00	NaN	NaN	12.5	4.4	9.9	NaN	7.8	6.3	11.7	8.6	—	4.8	5.7	6.0	NaN	5.0	5.0	5.8	5.3	NaN	4.7
2014-01-01 01:00:00	NaN	NaN	12.9	3.7	9.0	NaN	6.2	7.0	11.7	5.4	—	4.3	6.3	5.0	NaN	6.0	6.0	6.2	5.3	NaN	5.2
2014-01-01 02:00:00	NaN	NaN	11.9	3.1	10.8	6.9	6.3	7.6	11.4	6.9	—	4.6	6.9	8.1	NaN	5.3	5.1	8.3	4.5	NaN	5.8
2014-01-01 03:00:00	NaN	NaN	12.7	3.9	9.5	8.1	6.4	6.7	12.1	7.9	—	6.4	6.0	9.2	NaN	7.3	5.4	6.3	4.8	NaN	4.7
2014-01-01 04:00:00	NaN	NaN	14.6	4.6	11.0	8.0	8.5	6.8	13.3	8.3	—	6.5	6.6	7.5	NaN	7.1	6.1	7.4	5.9	NaN	5.5

FIGURE 10 – wind_velocity_df

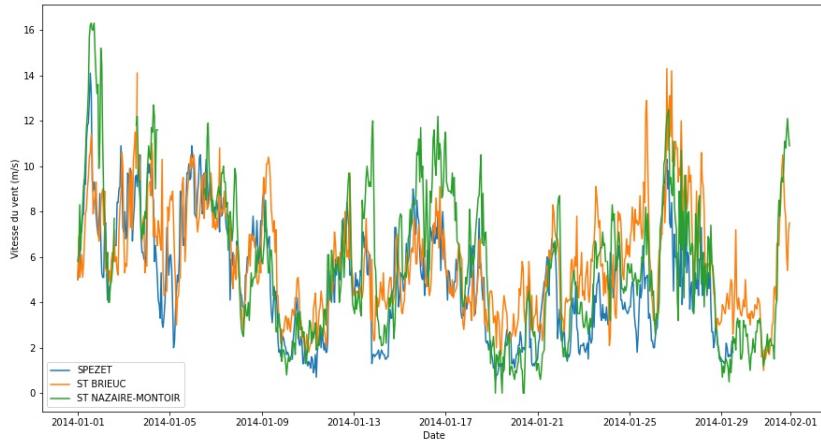


FIGURE 11 – Évolution du vent pour SPEZET, ST BRIEUC & ST NAZaire

station_name	ARZAL	AURAY	BELLE ILE LE TALUT	BIGNAN	BREST- GUIPAVAS	BRIGNOGAN	DINARD	GUERANDE	ILE DE GROIX	ILE DE BREHAT	—	SAINTE- CAST LE G	SARZEAU SA	SIBIRI SA	SIZUN	SPEZET	ST BRIEUC	ST NAZaire- Montoir	ST- SEGALS THEIX	VANNES SENE	
date																					
2014-01-01 00:00:00	9.7	10.3	11.3	7.1	9.7	NaN	6.8	10.4	9.0	7.0	—	6.1	9.8	7.4	8.5	8.3	7.5	9.0	9.0	9.3	9.7
2014-01-01 01:00:00	9.8	10.4	10.3	7.6	0.6	9.2	6.5	10.6	10.0	7.0	—	6.3	9.9	8.2	8.3	8.6	7.0	9.1	9.5	9.1	9.9
2014-01-01 02:00:00	9.7	10.0	11.2	7.7	9.1	9.4	6.5	10.2	9.5	7.8	—	6.4	9.2	8.5	8.5	8.6	7.5	9.5	9.4	9.4	9.6
2014-01-01 03:00:00	9.4	10.4	11.4	7.9	9.7	10.1	7.1	10.6	10.4	8.3	—	7.0	10.1	8.5	8.7	8.9	7.2	9.0	9.9	8.9	9.8
2014-01-01 04:00:00	9.8	10.8	11.4	8.8	9.8	10.2	7.6	10.6	10.8	8.9	—	7.7	10.5	9.1	9.1	9.4	7.6	9.5	10.0	9.6	10.2

FIGURE 12 – temperature_df

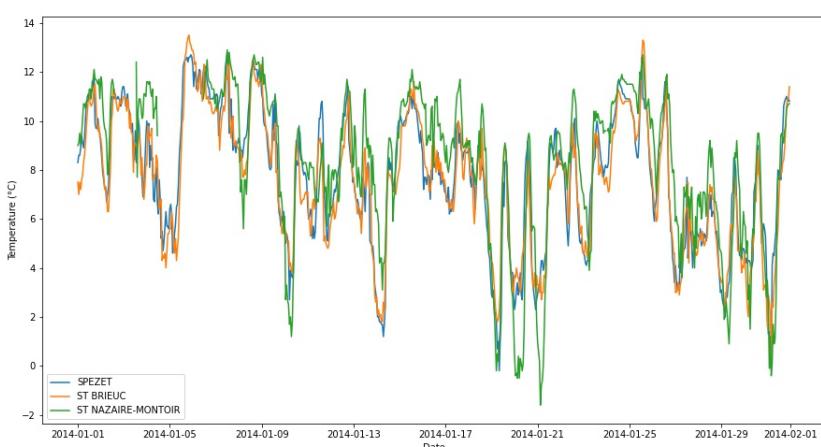


FIGURE 13 – Évolution de la température pour SPEZET, ST BRIEUC & ST NAZaire

Nous nous intéresserons par la suite, non pas à l'évolution de ces variables dans le temps mais à l'évolution spatiale à une date donnée.

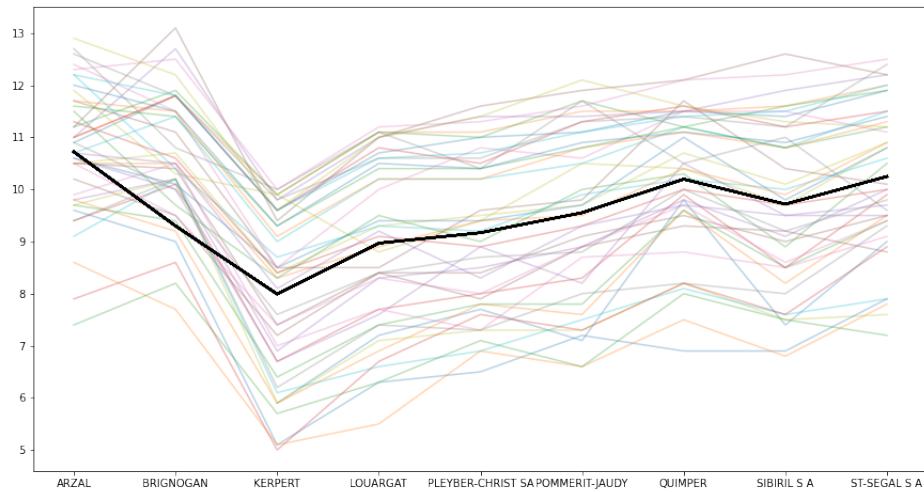


FIGURE 14 – Évolution de la température sur 9 stations & moyenne des températures (en noir) par station

5.1.3 Création du Graphe

Le package **Contextily** permet de récupérer des cartes en mosaïques sur internet (les délimitations peuvent être passées en WGS84 (EPSG :4326)). L'affichage peut se superposer au figure de **matplotlib**. Grâce au coordonnées GPS du dataset on peut donc superposer 2 figures : La carte de la Bretagne et les stations météorologiques.

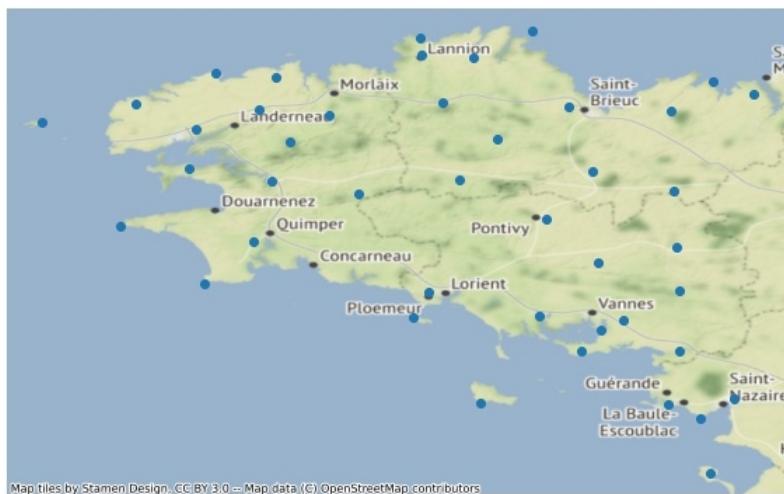


FIGURE 15 – Carte des stations météorologiques

Nous pouvons maintenant utiliser ces points comme noeuds de notre graphe. La création des arêtes est plus compliquée. En effet, connecter tous les noeuds entre eux créer un grand nombre d'arêtes. De plus, il serait absurde de considérer 2 stations très éloignées

géographiquement comme voisines dans notre graphe.

On se propose alors d'utiliser un noyau gaussien pour construire les arêtes en ajoutant un poids sur la distance qui sépare 2 stations.

On pose

$$W_{i,j} = \begin{cases} \exp\left(\frac{\|c_i - c_j\|^2}{\sigma^2}\right) \text{ si } \exp\left(\frac{\|c_i - c_j\|^2}{\sigma^2}\right) > \lambda \\ 0 \text{ sinon} \end{cases}$$

avec les c_i les positions des stations, σ le paramètre de largeur de bande et λ un seuil choisi arbitrairement.

Plus λ est proche de 0, plus notre graphe aura un grand nombre d'arêtes et réciproquement plus λ est proche de 1 moins notre graphe aura d'arêtes. On fixera λ à 0.85.

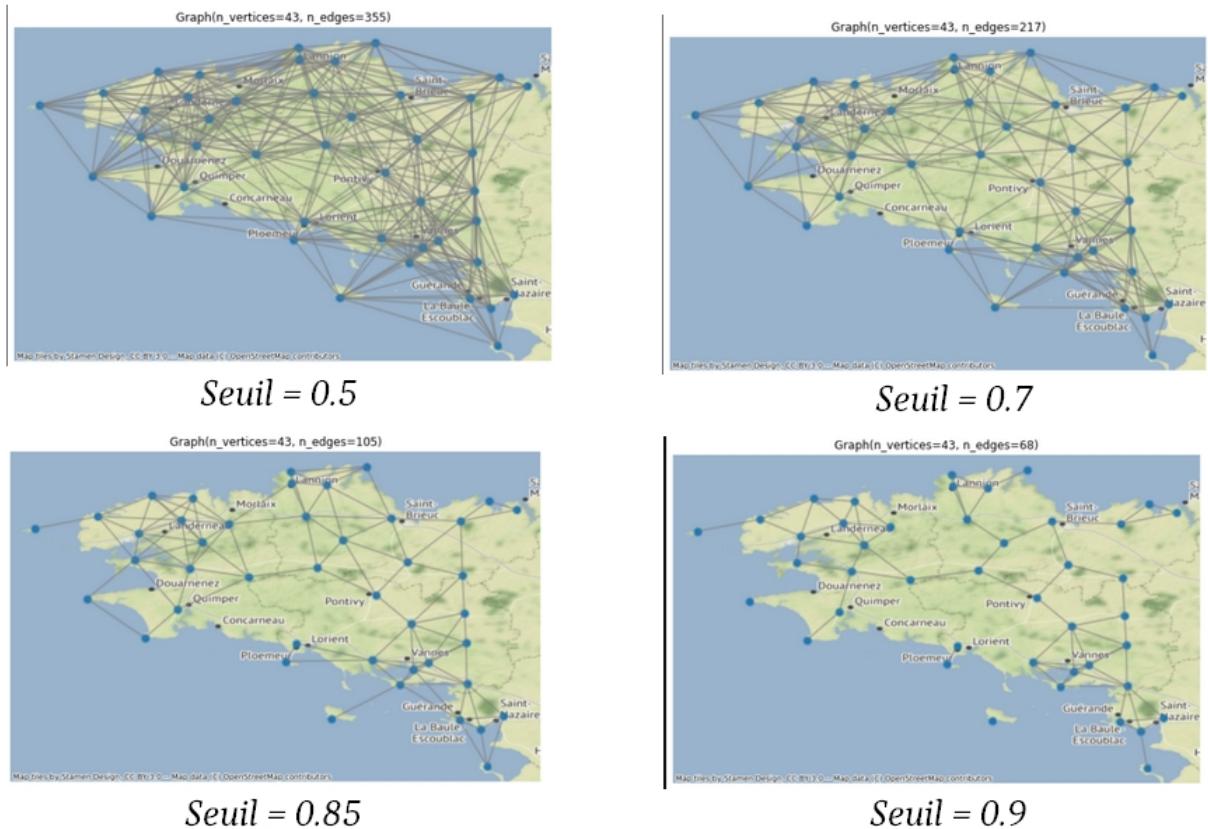


FIGURE 16 – Graphes différentes valeurs de λ

5.2 Étude sur les signaux

Notre objectif est de partitionner en clusters les stations de la Bretagne, nous cherchons donc à approcher le signal brut par un signal constant par morceaux ($k = 0$). Pour visualiser notre modèle, on prendra le signal des températures relevées le 1/1/2014 à 03h00.

5.2.1 Apprentissage constant par morceaux

Considérons le signal brut du 1/1/2014 à 03h00, on cherche à l'approcher par un signal constant par morceaux. Dans un premier temps, on calcule λ_{max} (ici $\lambda_{max} = 2.35$) et on fixe $\lambda < \lambda_{max}$.

La valeur minimale du signal est de 6.7 et la valeur maximale est de 11.5, la valeur moyenne est 9.2. L'ordre de grandeur des valeurs étant assez faible, on fixera la *précision* à 1e-4.

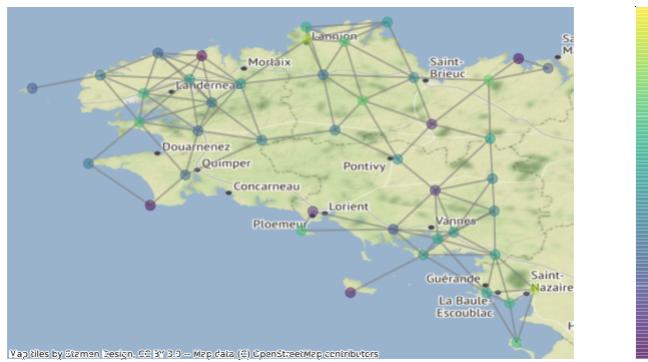


FIGURE 17 – Signal brut du 1/1/2014 03h00

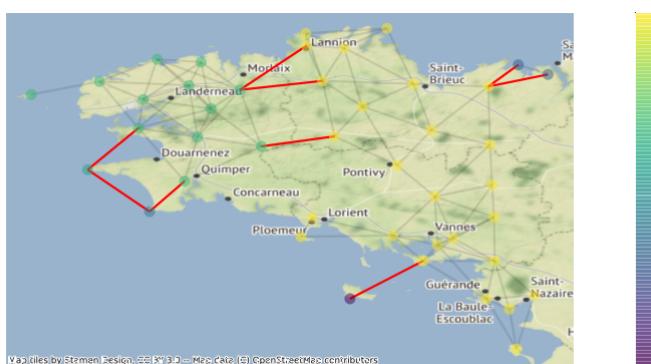
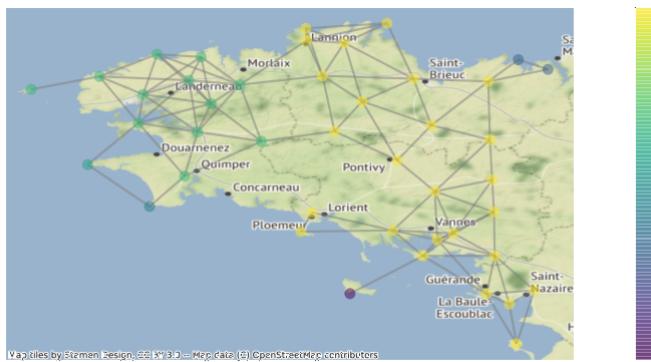


FIGURE 18 – Signal approché (avec et sans ruptures)

On remarque que notre signal peut être approché par un signal constant par morceaux, de 6 valeurs différentes. Cela se traduit par les 6 couleurs différentes présentes sur le graphe. Nous pouvons ainsi analyser les structures spatiales de la Bretagne vis-à-vis de la température en étudiant chaque relevé de température sur le mois de janvier 2014.

5.3 Calibration

5.3.1 λ

L'hyperparamètre λ et λ_{max} sont inhérents à chaque signal. Nous souhaitons, pour suivre une régularité des signaux appris au cours du temps, fixé des valeurs de λ qui soient cohérentes, qui ne créent pas d'erreur ou qui ne dépassent pas λ_{max} .

Par exemple, pour les données de température :

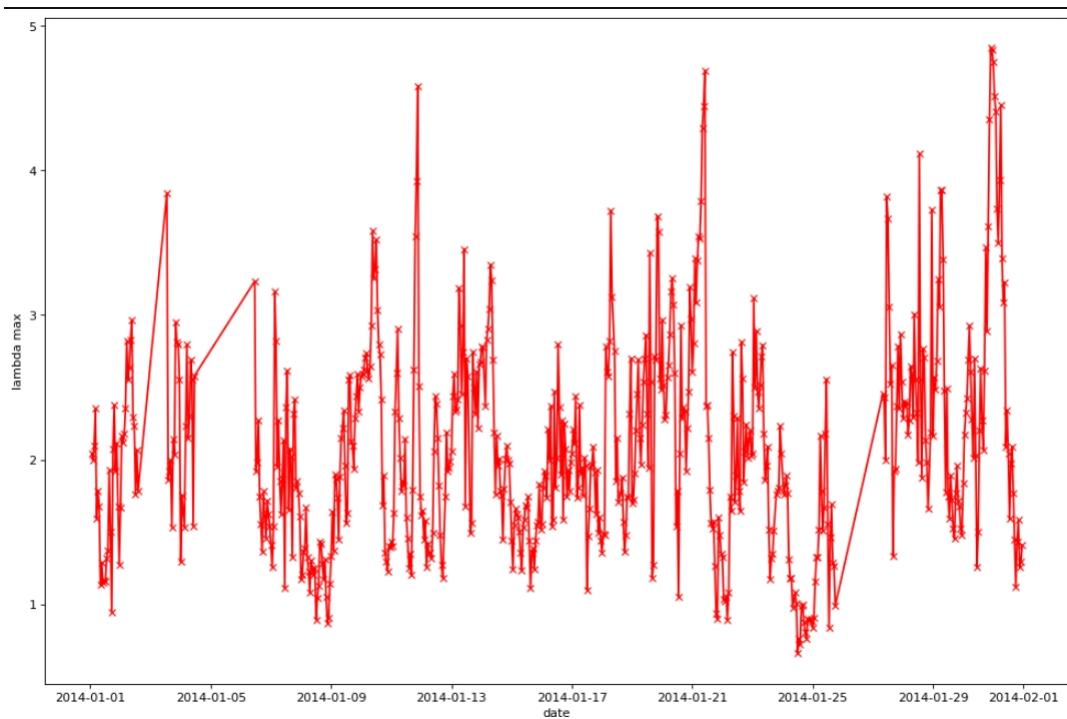


FIGURE 19 – Évolution du λ_{max} au cours du temps

On remarque que λ_{max} est très bas par moment, fixer une valeur absolue pour tous les signaux semble donc très compliqué.

On fixera pour cela, pour chaque signal, λ à un pourcentage du λ_{max} correspondant.

5.3.2 Regroupement en clusters

On regroupe les noeuds de mêmes valeurs du signal appris en clusters. L'évolution du nombre de clusters à cours du temps nous permet d'étudier la régularité du signal.

L'objectif est que le nombre de clusters soit plus ou moins le même au cours du temps et qu'il ne soit pas trop grand (on veut éviter d'avoir le même nombre de clusters que de noeuds).

Pour calculer le nombre de clusters d'un graphe, on détermine le nombre de composantes connexes présentes dans celui ci. Pour se faire, on calcule la multiplicité de la valeur propre 0 du Laplacien du graphe (qui est égale au nombre de composante connexe

du graphe).

On trace ainsi le nombre de clusters moyen de tous les signaux en fonction du pourcentage de λ_{max} utilisé dans notre modèle.

On peut ainsi choisir le pourcentage correspondant au nombre de clusters (en moyenne) que l'on souhaite avoir.

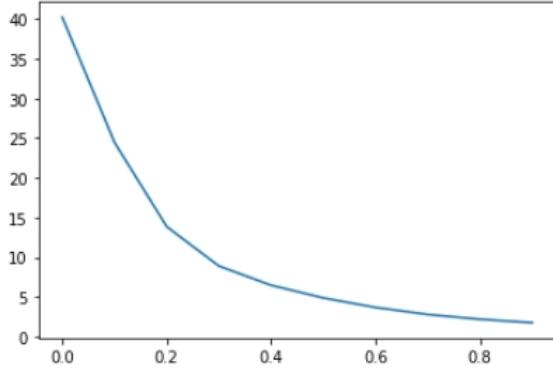


FIGURE 20 – Nombre de clusters moyen (des signaux de température) en fonction du pourcentage de λ_{max}

On souhaite partitionner spatialement la Bretagne en 6 ou 7 clusters. On choisit donc $\lambda = 0.4 * \lambda_{max}$.

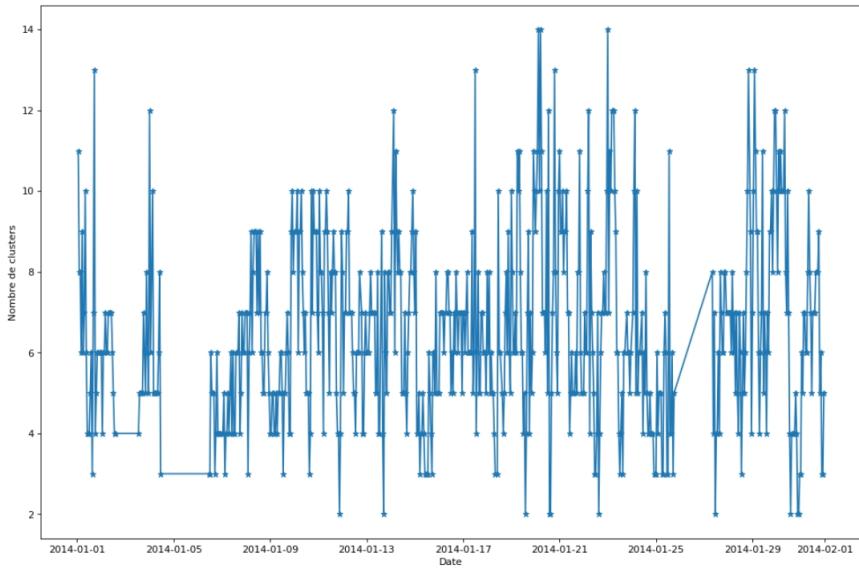


FIGURE 21 – Nombre de clusters au cours du temps ($\lambda = 0.4\lambda_{max}$)

On déterminer ensuite le signal moyen des signaux appris (avec $\lambda = 0.4 \lambda_{max}$).

On observe donc, qu'en moyenne, notre signal sur graphe semble être découpé en 6 ou 7 zones : La partie centrale ouest et la partie centrale est qui possèdent presque les mêmes températures, la zone sud-est liée à la Loire Atlantique, la zone nord est vers St Malo, et les grandes îles (Belle-Île au sud et Ouessant à l'est).

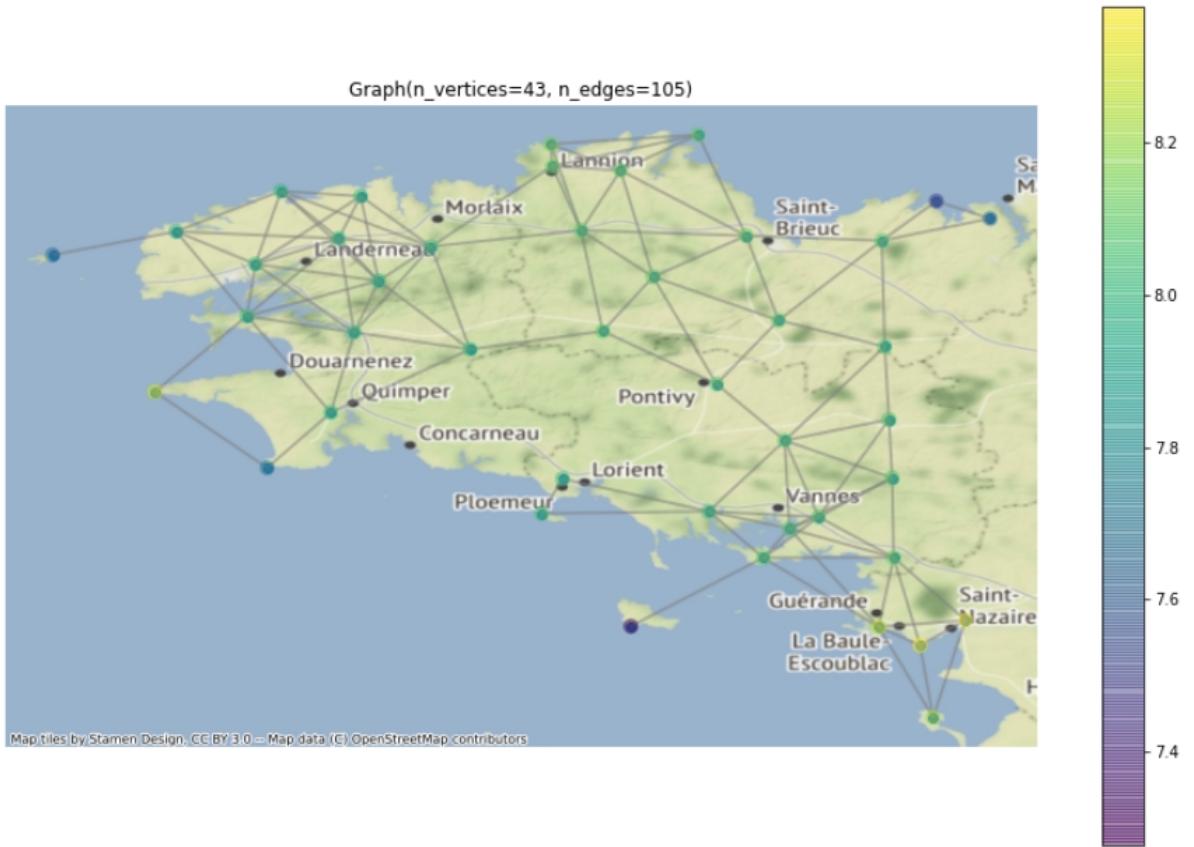


FIGURE 22 – Signal moyen des températures apprises au cours du temps

5.4 Smoothness

Afin de vérifier si notre modèle est régulier on calculera la smoothness s du signal :

$$s = \hat{\beta}^T L \hat{\beta}$$

avec L le laplacien du graphe.

On compare la smoothness sur le signal brut et sur le signal appris. On observe que le signal appris est plus régulier dans l'ensemble, cela nous montre que notre modèle réduit l'irrégularité du signal brut. Pour autant on observe tout de même les mêmes pics et parties stables, il est important de conserver la nature du signal.

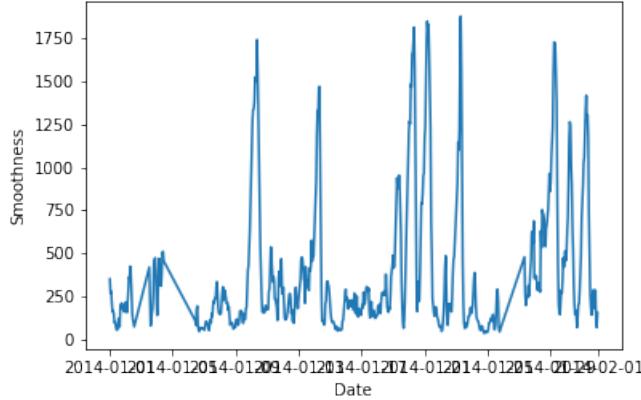


FIGURE 23 – Smoothness du signal brut

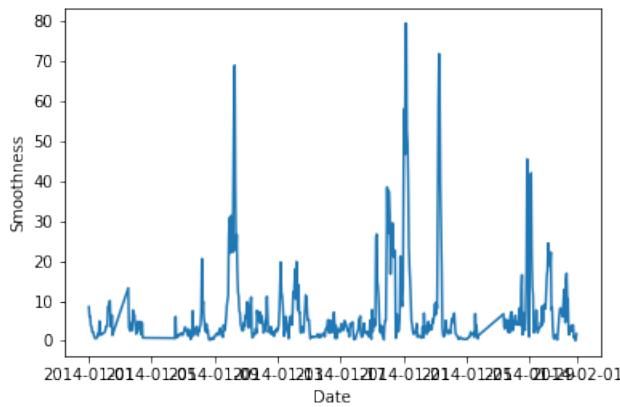


FIGURE 24 – Smoothness du signal appris

5.5 Interpolation de noeuds

L'objectif du stage est de pouvoir déterminer comment simplifier un signal sur graphe sans perdre d'information, on souhaite donc trouver, si possible, un ou des noeuds qui peuvent être enlevés de notre modèle sans que cette action ne le perturbe.

L'information contenu dans les voisins d'un noeud va nous aider à savoir si le dit noeud est supprimable. Nous allons donc supprimer un à un chaque noeud de notre graphe et essayer, de deux manières, de retrouver la valeur supprimée.

5.5.1 Approche par moyenne empirique

Une manière naïve de restituer un noeud est de calculer la valeur moyenne des noeuds adjacents au noeud absent.

$$\hat{x}_j = \frac{1}{n_j} \sum_{i=1}^n y_i e_{ij}$$

5.5.2 Approche par moyenne des prédictions

La seconde méthode consiste à utiliser le modèle constant par morceaux que l'on a générer. En effet, dans le cas où le noeud est au milieu d'un plateau de valeur, la valeur

restituée est la valeur du plateau. Dans le cas où le noeud possède des voisins de valeurs différentes, le noeud est sur un rupture du modèle, il prendra alors une valeur de transition entre les plateaux.

$$\hat{x}_j = \frac{1}{n_j} \sum_{i=1}^n \beta_i e_{ij}$$

5.6 Simplification du graphe

Afin d'estimer la performance de la restitution d'un noeud, on calculera l'erreur $\epsilon = \text{valeur brut du signal} - \text{valeur restituée}$. Pour chaque signal, on supprime un à un chaque noeud et on calcule l'erreur correspondante et on note le noeud avec l'erreur la plus faible.

On numérotera les noeuds de 0 à 42 par la suite :



FIGURE 25 – Graphe avec noeuds numérotés

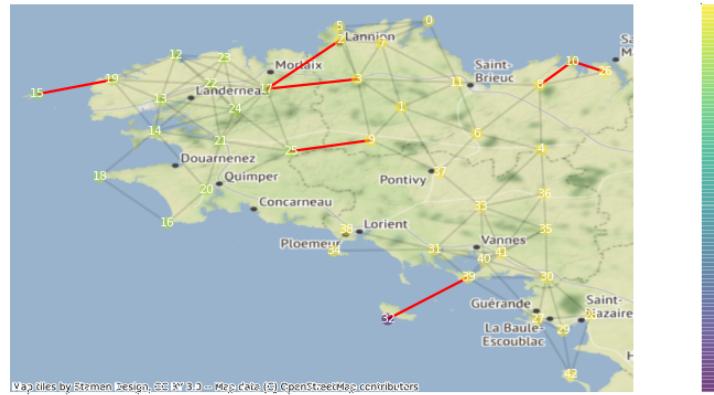


FIGURE 26 – Graphe appris avec noeuds numérotés

En parcourant tous les signaux et notant le noeuds avec l'erreur la plus faible, pour l'approche naïve on obtient la Figure 27 et pour l'approche par moyenne des prédictions on obtient la Figure 28.

Les deux méthodes désignent (presque) les mêmes noeuds : noeuds 8, 28 et 42. La méthode par moyenne des prédictions désigne, en plus, le noeud 35.

On remarque que ces noeuds sont situés sur les bords du graphe. Étant donné qu'ils sont connectés à moins de noeuds que la plupart des autres noeuds, ils ne représentent

pas une grosse rupture avec leurs noeuds voisins (comme on pourrait avoir avec le noeud 32 isolé mais on observe une différence notable de valeur entre lui et son voisin).

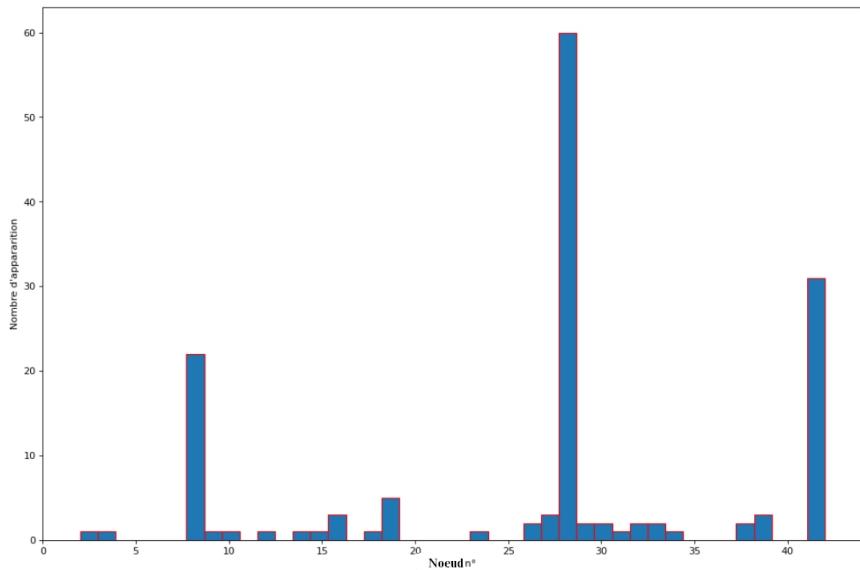


FIGURE 27 – Histogramme des noeuds restitués par méthode naïve

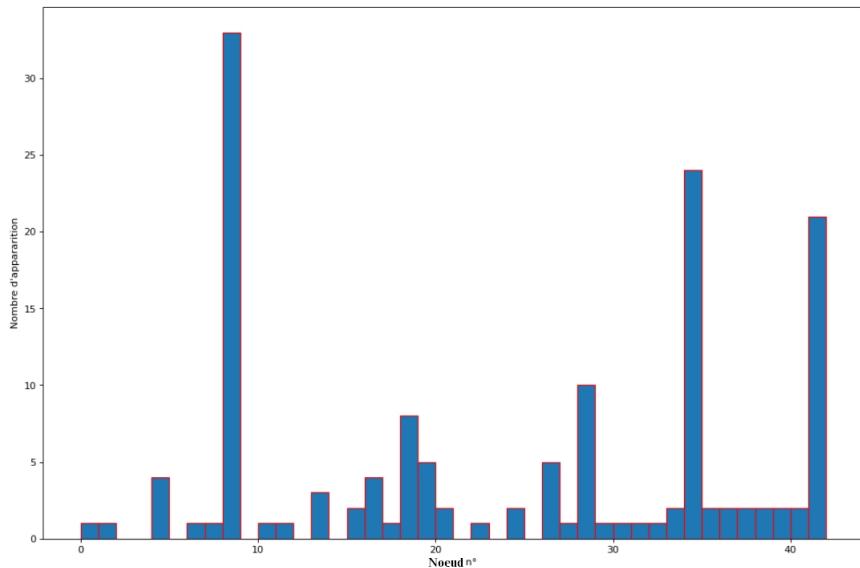


FIGURE 28 – Histogramme des noeuds restitués par moyenne des prédictions

6 Conclusion

Pour conclure, l'étude que nous avons menée s'inspire du trend filtering généralisé aux graphes, afin de découvrir quantitativement les structures spatiales régulières de la Bretagne, grâce notamment aux données de température. Le modèle utilisé est très complexe et que les paramètres inhérents à chaque signal doivent être correctement calibré pour obtenir des résultats cohérents.

Nous avons observé, grâce au modèle mis en place, que la Bretagne pouvait être regroupée en 6 à 7 zones en fonction de la température du mois de janvier 2014. Aussi, nous avons étudié la possibilité de simplifier le graphe en supprimant, en essayant d'interpoler la valeur d'un noeud avec notre modèle, des noeuds portant le moins d'information du signal. Néanmoins, un travail supplémentaire peut être réalisé afin de corriger l'interpolation de noeud. En effet, nous avons travaillé avec la moyenne des valeurs des noeuds voisins du signal brut et appris. D'autres méthodes, plus efficaces, doivent pouvoir être mises en place. Aussi, notre problématique s'est posé sur l'étude du signal sur graphe de manière univarié, il serait pertinent de continuer cette étude et la transposer dans le cadre multivarié en utilisant les différentes données mis à disposition par le data set.

Ce stage s'est basé essentiellement sur l'étude du trend filtering sur graphe pour l'apprentissage constant par morceaux, notre modèle peut toutefois être adapté pour n'importe quel ordre (linéaire, quadratique, cubique, etc ...). Ainsi il peut être utilisé dans de nombreux domaines comme l'étude du mouvement des foules, des populations, de la marégraphie, des maladies, etc. ou plus généralement tout système pouvant être approché par une série temporelle.

Ce fut un réel plaisir de réaliser ce stage. En effet, souhaitant suivre une thèse l'an prochain, j'ai pu découvrir, à travers les articles de recherche et de thèse que j'ai pu étudier, ce monde. Malgré le cadre du stage en distanciel, j'ai eu la chance de pouvoir bien m'investir et comprendre toutes les nuances du stage grâce à M. Charles TRUONG et je le remercie pour cela.

A Annexe : Développement durable et responsabilité sociétale

A.1 Développement durable

Durant ce stage, les conditions sanitaires actuelles nous ont contraint à travailler en distanciel. Étant logé sur Évry, le trajet du domicile au lieu de stage se trouvant à Saclay doit se faire par voiture ou transport en commun. Cette absence de déplacement annule donc l'empreinte carbone.

Chaque trajet aller retour représente plus de 6 kg de CO₂ rejeté. Donc cela représente au moins 648 kg de CO₂ rejeté pour l'entièreté du stage.

De plus, notre travail nécessite essentiellement notre ordinateur personnel et est donc peu coûteux en matériel.

Annexes

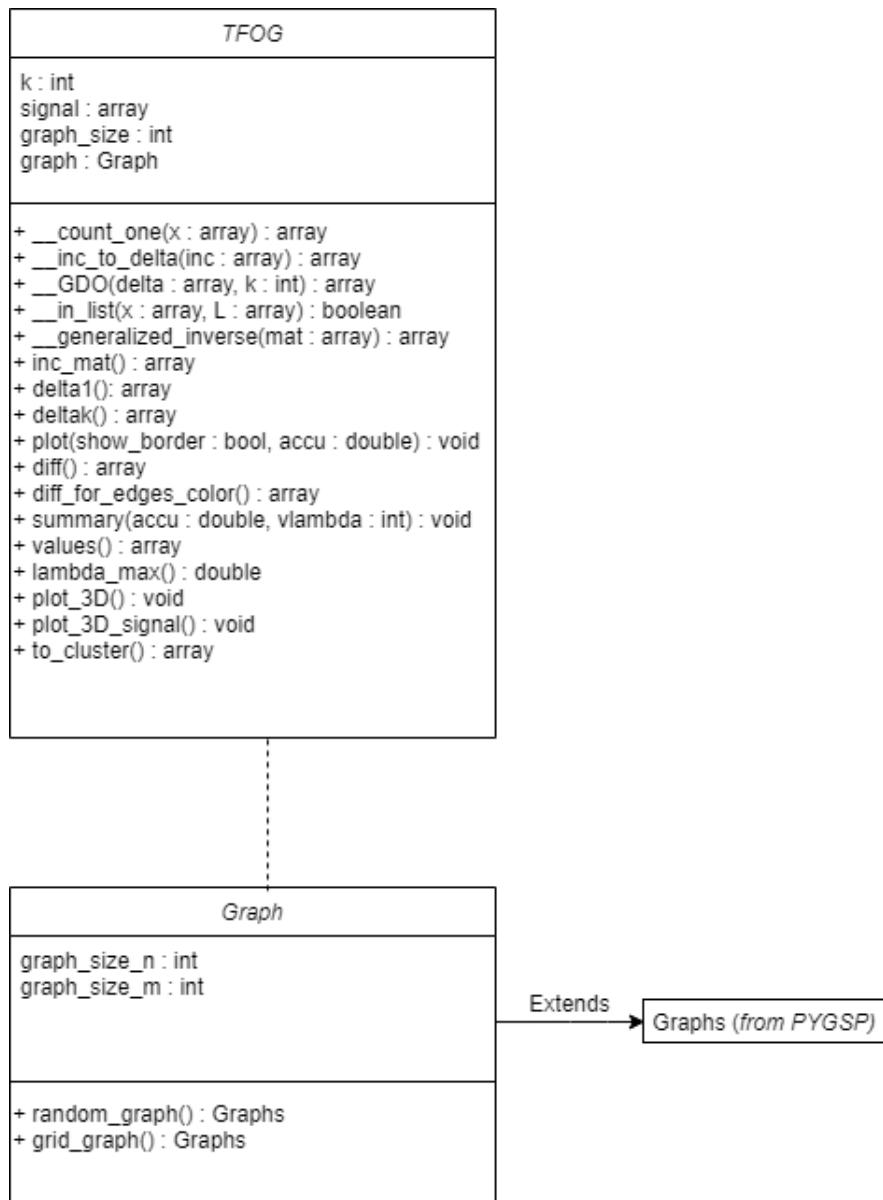


FIGURE 29 – Diagrammes des classes

Descriptif	Mnémonique	type	unité
Paramètres standard			
Indicatif INSEE	station	numer_sta	car
Indicatif	OMM station	id_omm	int
Date	date	date	car
Point de rosée	td	réel	K
Température	t	réel	K
Température maximale de l'air	tx	réel	K
Température minimale de l'air	tn	réel	K
Humidité	u	int	%
Humidité maximale	ux	int	%
Humidité minimale	un	int	%
Direction du vent moyen 10 mn	dd	int	degré
Vitesse du vent moyen 10 mn	ff	réel	m/s
Direction du vent moyen maximal	dxy	int	degré
Vitesse maximale du vent tmoyen	fxy	réel	m/s
Direction du vent instantané maximal	dxi	int	degré
Vitesse maximale du vent instantané	fxi	réel	m/s
Précipitations dans l'heure	rr1	réel	kg/m ²
Paramètres selon instrumentation spécifique			
Température à -10 cm	t ₁₀	réel	K
Température à -20 cm	t ₂₀	réel	K
Température à -50 cm	t ₅₀	réel	K
Température à -100 cm	t ₁₀₀	réel	K
Visibilité horizontale	vv	réel	m
Etat du sol	etat_sol	int code	
Hauteur totale de la couche de neige	sss	réel	m
Nebulosité totale	n	réel	%
Durée insolation	insolh	int	mn
Rayonnement global	ray_glo01	réel	J/m ²
Pression station	pres	int	Pa
Pression au niveau mer	pmer	int	Pa

FIGURE 30 – Data set

Liste des figures

1	Représentation du trend filtering d'ordre linéaire ($k = 1$)	5
2	Exemples de graphes	6
3	Signal sur graphe circulaire	6
4	Signal sur graphe "grille"	7
5	Signal linéaire sur graphe grille	7
6	Signal approché constant par morceaux	7
7	Visualisation des ruptures	8
8	Signal appris pour $\lambda = 78$ et $\lambda \geq \lambda_{max}$	9
9	stations_gdf	12
10	wind_velocity_df	13
11	Évolution du vent pour SPEZET, ST BRIEUC & ST NAZAIRE	13
12	temperature_df	13
13	Évolution de la température pour SPEZET, ST BRIEUC & ST NAZAIRE	13
14	Évolution de la température sur 9 stations & moyenne des températures (en noir) par station	14
15	Carte des stations météorologiques	14
16	Graphes différentes valeurs de λ	15
17	Signal brut du 1/1/2014 03h00	16
18	Signal approché (avec et sans ruptures)	16
19	Évolution du λ_{max} au cours du temps	17
20	Nombre de clusters moyen (des signaux de température) en fonction du pourcentage de λ_{max}	18
21	Nombre de clusters au cours du temps ($\lambda = 0.4\lambda_{max}$)	18
22	Signal moyen des températures apprises au cours du temps	19
23	Smoothness du signal brut	20
24	Smoothness du signal appris	20
25	Graphe avec noeuds numérotés	21
26	Graphe appris avec noeuds numérotés	21
27	Histogramme des noeuds restitués par méthode naïve	22
28	Histogramme des noeuds restitués par moyenne des prédictions	22
29	Diagrammes des classes	25
30	Data set	26

Bibliographie

Yu-Xiang Wang, James Sharpnack, Alexander J. et Ryan J. Tibshirani Trend Filtering on graphs
<https://www.stat.cmu.edu/~ryantibs/papers/graphtf.pdf>

Seung-Jean Kim, Kwangmoo Koh, Stephen Boyd, Dmitry Gorinevsky Trend Filtering
https://web.stanford.edu/~boyd/papers/pdf/l1_trend_filter.pdf

Toward DataScience Introduction to TF <https://towardsdatascience.com/introduction-to-trend-filtering-with-applications-in-python-d69a58d23b2>

Optimization Online Optimization Online for datascience
http://www.optimization-online.org/DB_FILE/2007/09/1791.pdf

Stackexchange Lasso and Ridge tuning parameter scope
<https://stats.stackexchange.com/questions/210040/lasso-and-ridge-tuning-parameter-scope/283049283049>

CVXOPT CVXOPT Examples https://www=cvxpy.org/examples/applications/l1_trend_filter.html

PYGSP PYGSP tutorial <https://pygsp.readthedocs.io/en/stable/>

Nisheeth K. Vishnoi Laplacian problems & Algorithm <https://theory.epfl.ch/vishnoi/Lxb-Web.pdf>