Interpolaspline

rapport

CORBILLE Clément, DOUMBOUYA Mohamed, EL BOUCHOUARI Zakaria,
HEDDIA Bilel, PIASENTIN Béryl, RODET Amélys

Avril 2020

Table des matières

I
Introduction

Dans le cadre de notre troisième année de licence en mathématiques et informatique, un stage applicatif est obligatoire. L'objectif de ce stage est de nous préparer au milieu professionnel. Il nous initie au projet de groupe et à la relation avec Mr Perrier, chercheur à l'INRIA et client de notre travail durant quatre semaines.

Le projet consiste en la création d'une fonction interpolant des données ainsi qu'en la minimisation de l'impact de données dites "aberrantes" sur cette fonction. Nos objectifs sont donc de trouver la fonction de manière automatique, et de chercher plusieurs méthodes pour détecter les données aberrantes et les traiter.

Dans ce compte rendu, nous allons en premier vous décrire le projet, en particulier le sujet et l'organisation mise en place durant les trois semaines de réalisation. En seconde partie, nous vous présenterons les définitions sur lesquelles nous nous sommes basés, les différentes méthodes découvertes suite à nos recherches, mais aussi une comparaison des résultats produits par nos différents algorithmes. Nous expliquerons et illustrerons ensuite le fonctionnement de notre application, qui répond aux besoins du client, avant de conclure ce projet.

II
Description du projet

1 Sujet

En raison de l'utilisation intensive des données, la gestion des valeurs aberrantes a acquis une grande importance ces dernières années. La présence de valeurs aberrantes peut alors conduire à de faux résultats, ce qui peut inciter des décisions risquées. C'est pourquoi nous avons cherché durant ce projet des solutions susceptibles de gérer les données aberrantes.

L'objectif principal de ce projet est de chercher des méthodes qui gèrent ces points aberrants, lors d'une interpolation des données.

1.1 Splines

Nous allons, pour étudier la gestion des valeurs aberrantes, générer des valeurs et essayer de les interpoler ou de les approximer par une fonction (possédant certaines propriétés, comme la continuité). Cette fonction sera une spline, nous définirons ce terme plus loin dans le rapport.

La différence entre l'interpolation et l'approximation mérite d'être expliquée : l'interpolation est la construction d'une fonction qui passe par toutes nos données, tandis que la fonction construite par approximation ne passera pas forcement par tous nos points mais fera au mieux (cette seconde option est privilégiée en présence d'un nuage de points). Les figures suivantes illustrent la différence entre interpolation (Figure 1) et approximation (Figure 2).

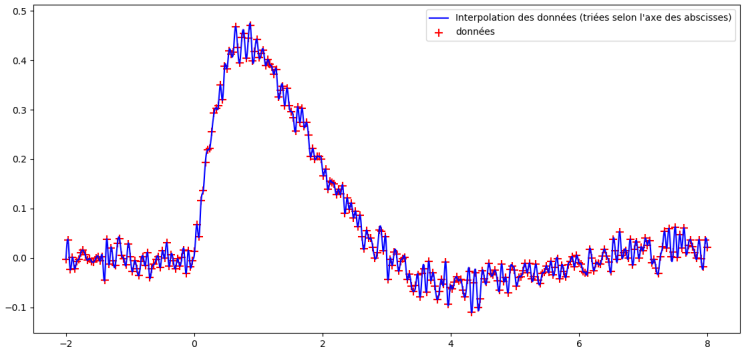


FIGURE 1 – Exemple d'interpolation sur un jeu de données quelconque

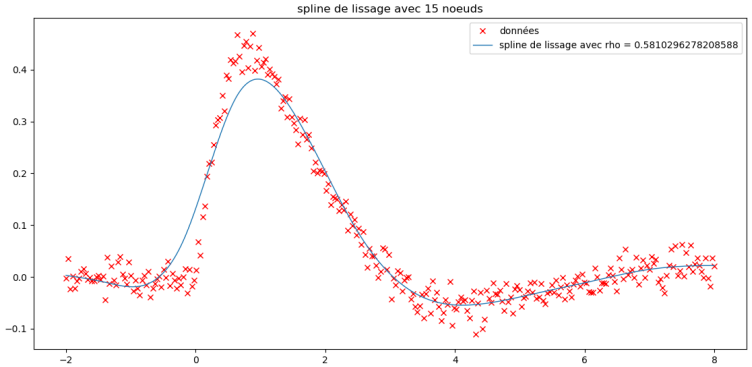


FIGURE 2 – Exemple d'approximation avec mêmes données

Cependant, l'interpolation et l'approximation ont un objectif commun : créer une fonction qui correspond le mieux possible aux données. C'est pourquoi nous allons nous permettre un abus de langage dans la suite de ce rapport : le terme interpolation sera le seul utilisé, même dans le cas d'approximation (pour les splines de lissage par exemple).

1.2 Données aberrantes

Nous avons parlé jusque là de données aberrantes sans les définir réellement. Nous allons ici discuter des différents sens existants, et expliciter la définition considérée pour le reste du rapport.

Les valeurs aberrantes peuvent être des données qui n'ont aucun sens comme par exemple une réponse absurde dans un questionnaire ou bien une erreur de mesure. On souhaite les ignorer totalement.

Les données aberrantes peuvent aussi être définies comme des valeurs qui se situent loin de la majorité des autres valeurs mais qui sont nécessaires pour l'étude, comme par exemple l'âge des étudiants du supérieur : une grande majorité aura un âge inférieur à 30 ans, mais il peut y en avoir quelques-uns ayant plus de 60 ans (même si cela est très rare) : nous voulons aussi intégrer ces étudiants dans les analyses. Ils constituent des données aberrantes, mais ce n'est pas une réponse absurde ou une erreur de mesure.

La définition que l'on retient pour la suite du rapport va être moins détaillées que les deux proposées précédemment, car nous générons des données aberrantes, mais nous n'en connaissons pas l'origine (Personne particulière ? Erreur de saisie ?). Pour nous, une donnée aberrante est donc définie comme une donnée distante des autres observations effectuées sur le même phénomène.

2 Organisation

L'organisation de ces trois semaines de stage applicatif a été planifiée lors de la première semaine du projet, en décembre, qui était plus amplement dédiée à la gestion de projet. Lors de cette semaine, nous avions conclu que Béryl serait notre chef de projet. Elle a veillé au mieux sur le bon déroulement de chaque tâche, et a facilité la communication au sein de l'équipe en plus de sa participation à certaines tâches.

Le premier jour a permis à tout le monde de se remettre dans le sujet et de se plonger dans leurs tâches respectives. Pour la suite, ces trois semaines de développement et de recherche ont permis à chaque membre du groupe d'être responsable du bon fonctionnement d'une ou de plusieurs tâches. Notre approche avant ces trois semaines était que les tâches devaient se terminer dans les délais, fixés par nous-mêmes. Mais nous nous sommes logiquement rendus compte qu'avec notre faible expérience dans l'organisation de travaux de groupes et les nombreux facteurs (humains et matériels) qui interviennent, la planification allait être remaniée maintes et maintes fois : nous pouvons voir avec les annexes 2 et 3 la différence entre la répartition des tâches originelle et finale. La correspondance entre les numéros et les contenus se trouve à l'annexe 1.

Nous avons donc rapidement remarqué que la planification que nous avions faite de ce projet ne serait pas respectée. Mais grâce à notre communication, nous avons pu limiter les problèmes de dépendances des tâches et avancer correctement, de manière à effectuer le plus grand nombre possible de nos tâches avant la fin de ces trois semaines.

Notre communication textuelle et orale en temps de confinement s'est principalement basée sur le logiciel Discord. Sa facilité d'utilisation nous a permis de tenir régulièrement informé le

groupe sur nos avancées respectives, et de faire quelques petites réunions entre nous. Ce logiciel nous a également servi à contacter le client lors de questions à propos de ses attentes précises.

Le reste de la communication au sein de l'équipe s'est quant à elle basée sur plusieurs logiciels, comme Git (GitKraken, Github et Git) pour les fichiers d'implémentation, Overleaf pour le rapport, et Google Documents pour les documents d'organisation.

3 Éléments fournis par le client

Ce projet est un projet de recherche. Cependant, l'objectif ne se limite pas au traitement de données aberrantes : nous devons nous baser sur les besoins du client car il est le premier concerné par ce projet. Mr Perrier nous a fourni un générateur de signaux bruités : notre objectif est de réussir à reconstruire ces signaux malgré le bruit.

Le programme fourni est en python. Il possède plusieurs fonctions. Ce programme nous permet de générer un ensemble de points en discrétisant un signal (stationnaire ou non) et en lui appliquant ensuite un bruit afin de faire apparaître des points aberrants.

Les signaux stationnaires sont des courbes lisses tandis que les signaux non stationnaires sont crénelés.

III
Réalisation

Pour interpoler les données, il faut choisir un modèle d'interpolation. Nous allons en évoquer quelques-uns.

Chaque modèle nécessite d'estimer les valeurs prises par une fonction continue entre deux points déterminés. La fonction et ses propriétés dépendent du modèle choisi.

L'interpolation linaire est considérée comme le modèle le plus simple. La fonction recherchée est une droite, dont l'équation est donc de la forme $f(x) = a * x + b$ avec a et b des paramètres réels. L'interpolation ne peut être faite qu'avec des points alignés. Si ce n'est pas le cas, il faut faire de l'approximation. Cette méthode est facile et simple à utiliser grâce au faible nombre de paramètres, mais sa précision dépend beaucoup des données : si elles sont très dispersées et loin d'un alignement, la droite ne sera pas très représentative des données.

Un second type d'interpolation assez connu est l'interpolation polynomiale. L'interpolation polynomiale consiste à interpoler les données par un polynôme d'un degré fixé. Le polynôme interpolant précisément des données est unique. L'interpolation de Lagrange est l'une des méthodes les plus célèbres pour l'interpolation polynomiale : c'est une version simple, qui impose simplement le passage du polynôme par tous les points donnés (x_i, y_i). Étant donné un ensemble de n+1 points on cherche à trouver un polynôme P de degré n au plus qui vérifie $P(xi) = y_i, i = 0...n$. Malheureusement, le résultat n'est pas toujours à la hauteur des espérances car lorsqu'il s'agit d'un modèle avec plusieurs points à interpoler, les polynômes obtenus vont avoir de très grands degrés : cela engendre des oscillations. Cette problématique nous a poussés à chercher d'autres solutions.

1 Splines

Soit n le nombre de données à interpoler. Une spline est une fonction définie par morceaux. Les points délimitant les morceaux sont appelés les noeuds de la spline. Chaque morceau est

un polynôme. Il y aurait, avec cette définition, une infinité de fonctions possible pour un seul jeu de données. C'est pourquoi il y a une contrainte supplémentaire : la fonction globale doit être C^k, avec $k \in \mathbb{N}$. Cela signifie que la fonction qui interpole les données doit être continue, et toutes ses dérivées jusqu'à la $k^{ième}$ inclue doivent l'être également.

L'interpolation par une spline constitue donc une alternative à l'interpolation par un polynôme de haut degré car les polynômes de la splines peuvent être de bas degré.

1.1 Splines cubiques C^2 naturelles

Les splines cubiques sont composées de polynômes de degré trois. Nous allons uniquement considérer les splines cubiques C^2 dans ce projet.

Soit m le nombre de noeuds de la spline, ici confondus avec les données que l'on souhaite interpoler. Ils délimitent $m-1$ intervalles, et donc $m-1$ polynômes. Chaque polynôme de degré trois possède 4 inconnues, ce qui donne $4(m-1) = 4m-4$ inconnues. Étudions maintenant les contraintes :

- $C^0 \Rightarrow 2m-2$ contraintes.
  En effet, les noeuds possèdent chacun une valeur. Chaque polynôme passe par 2 noeuds, donc possède deux contraintes : au total, la condition C^0 engendre $(m-1)*2 = 2m-2$ contraintes.
- Dérivée première continue $\Rightarrow m-2$ contraintes.
  En effet, chaque noeud interne (on exclue le premier et le dernier noeud) doit avoir la même dérivée à droite et à gauche, ce qui donne une contrainte (la valeur de la dérivée) par noeud. Il y a $m-2$ noeuds internes, donc la continuité de la dérivée première engendre $m-2$ contraintes.
- Dérivée seconde continue $\Rightarrow m-2$ contraintes.
  Le raisonnement est identique à celui de la dérivée première.

On obtient au total $(2m-2)+(m-2)+(m-2) = 4m-6$ contraintes. Cela nous donne $(4m-4)-(4m-6) = 2$ degrés de liberté, ce qui implique une infinité de fonctions solutions.

Pour avoir une unique spline cubique C^2 qui interpole les m noeuds associés à un jeu de données, il ne faut plus avoir de degré de liberté. Une possibilité est de décréter les dérivées premières aux extrémités nulles : cela ajoute deux contraintes, ce qui enlève les deux degrés de liberté laissés par la définition des splines cubiques C^2. Lorsque c'est cette solution qui est choisie, la spline cubique C^2 recherchée est dite "naturelle".

Nous avons également implémenté les splines naturelles pour des données à deux dimensions. Nous l'avons fait avec la méthode intuitive, c'est à dire en créant une spline A pour les abscisses et une spline B pour les ordonnées, toutes les deux en fonction d'un paramètre t, puis en traçant la spline dont les abscisses sont les valeurs de A et les ordonnées sont les valeurs de B.

1.2 Splines cubiques C^2 de lissage

Une spline de lissage permet de satisfaire un compromis entre la fidélité aux observations bruyantes et le lissage de la spline ajustée.

Les splines de lissage sont des splines cubiques dont chaque polynôme de degré trois est une approximation des données se trouvant sur son intervalle de définition. Cette spline ne passera pas (dans la plupart des cas) par tous les points. Elle minimise en revanche une quantité liée à la distance entre les données et la spline. En général, la spline approximant les données cherche

à minimiser l'erreur au carré : c'est l'approximation aux moindres carrés. Ces splines de lissage permettent d'éviter les oscillations qui seraient présentes avec une spline naturelle passant par tous les points, provoquées avec un nombre de données très grand.

Notre premier but, avant d'entrer dans le vif du projet, était de reprendre les programmes écrits pendant les cours d'algèbre linéaire pour le graphique et la CAO (Conception assistée par ordinateur). Ces programmes étant adaptés à des énoncés de travaux pratiques, nous les avons donc repris pour les compléter et les adapter à nos besoins. De plus amples d'informations sur ces travaux se trouvent dans le polycopié de cours et sur les énoncés fournis par Mr Biard aux étudiants L3 MI disponibles depuis sa page internet personnelle.

1.2.1 Répartition uniforme

Ce qui différencie les splines de lissage uniforme des autres splines de lissage, c'est la répartition des noeuds délimitant les différents polynômes cubiques : elle est uniforme. Cela signifie que les noeuds sont distribués de manière équidistante dans un intervalle déterminé, avec un pas strictement positif h $= x_{i+1} - x_i, i = 1, ..., n-1$ pour les n données $\{x_1, ...x_n\}$. La figure ci-dessous illustre la répartition uniforme de 15 noeuds (avec différentes splines de lissage, pour un unique jeu de données).

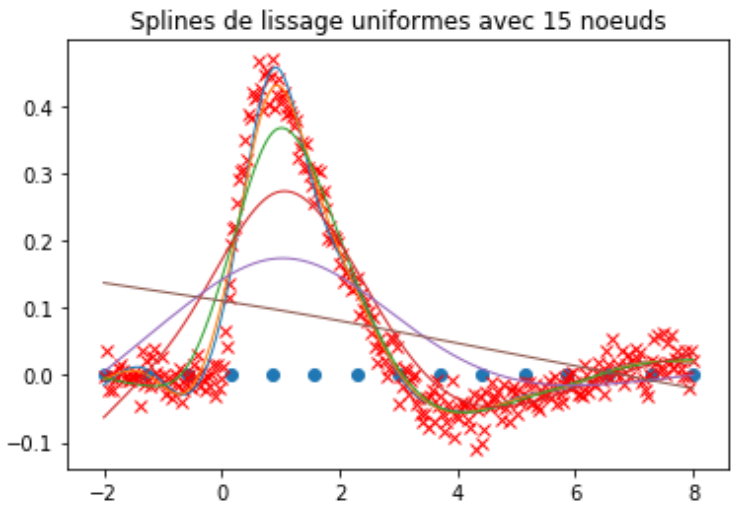


FIGURE 3 – Exemple de plusieurs courbes de lissages uniformes avec différents paramètres de lissage ρ

La spline est ensuite construite. Pour cela, il faut trouver la valeur des dérivées en chaque noeud. On applique ensuite le procédé d'interpolation de Hermite. Pour faire court, cette interpolation C^2 consiste en la construction d'un polynôme qui doit à la fois :

- Coïncider avec celles de la fonction déduite de l'échantillon aux points donnés
- Avoir une dérivée qui coïncide avec celle de la fonction déduite de l'échantillon aux points donnés

Comme dans notre cas nous considérons des splines cubiques C^2, cette interpolation s'applique également pour la dérivée secondaire. Pour plus d'informations sur la construction des splines de lissage, il faut se référer au cours de CAO.

1.2.2 Répartition non-uniforme

Lorsque nous souhaitons répartir les noeuds de manière non-uniforme, il faut calculer les matrices nécessaires. On s'inspire des calculs pour le cas uniforme effectués par M. Biard dans

son cours de CAO.

Soient $\{(u_k, z_k), k \in \mathbf{N} \mid u_i < u_j, \forall (i,j) \in \mathbf{R}^2, i < j\}$ nos données. Considérons les $n \in \mathbf{N}$ noeuds de lissage $\{x_k, k \in \mathbf{N},\ k < n \mid x_i < x_j, \forall (i,j) \in \mathbf{R}^2, i < j\}$ et l'espace $S[x_1, x_n]$ des splines naturelles associées à ces points.

On considère le problème d'optimisation suivant, qui est la minimisation d'une quantité :

$$Min_{s \in S[x_1,x_n]} E_{0,2}(s)$$

avec

$$E_{0,2}(s) = \sum_{k=1}^{N} (z_k - s(u_k))^2 + \rho \int_{x_1}^{x_n} [s''(t)]^2$$

ρ est le paramètre de lissage de la spline, comme dans le cas uniforme.

De la même manière que dans le cas uniforme (et avec les mêmes notations), on détermine les matrice A, R, S, M, N, $H_{0,3}$ et $H_{1,2}$. Commençons par les matrices A et R : on cherche, comme dans le cas uniforme, la relation entre y et y' pour une spline cubique naturelle C^2, qui peut être écrite de la forme Ay' = Ry. Nous avons les conditions $s''(x_1) = s''x_n = 0$ (spline naturelle) et $s''_{i-1}(x_i) = s''_i(x_i)$ (contact aux noeuds C^2).
Nous obtenons, en appliquant la même méthode que dans le cas uniforme, ces trois relations entre y et y' :

$$\begin{cases} 2\text{y'}_1 + y'_2 & = 3/h_1(y_2 - y_1) \\ \text{h}_i y'_{i-1} + 2(h_{i-1} + h_i)y'_i + h_{i-1}y'_{i+1} & = 3(\frac{-h_i}{h_{i-1}} y_{i-1} + (\frac{h_i}{h_{i-1}} - \frac{h_{i-1}}{h_i})y_i + \frac{h_{i-1}}{h_i} y_{i+1}) \\ , i = 2...n-1 & \\ \text{y'}_{n-1} + 2y'_n & = 3/h_{n-1}(y_n - y_{n-1}) \end{cases}$$

La relation entre y et y' étant de la forme Ay'= Ry, on obtient

$$A = \begin{pmatrix} 2 & 1 & 0 & 0 & ... & ... & ... & 0 & 0 \\ h_2 & 2(h_1+h_2) & h_1 & 0 & ... & ... & ... & 0 & 0 \\ \vdots & . & . & . & . & ... & . & . & . \\ \vdots & . & . & . & h_i & 2(h_{i-1}+h_i) & h_{i-1} & 0 & \vdots \\ . & & & & & & & & \\ . & . & . & . & . & ... & . & . & . \\ 0 & 0 & . & . & . & ... & . & 1 & 2 \end{pmatrix}$$

$$R = 3 \begin{pmatrix} -\frac{1}{h_1} & \frac{1}{h_1} & 0 & 0 & ... & ... & ... & 0 & 0 \\ \frac{h_2}{h_1} & \frac{h_2}{h_1} - \frac{h_1}{h_2} & \frac{h_1}{h_2} & 0 & ... & ... & ... & 0 & 0 \\ \vdots & . & . & . & . & ... & . & . & . \\ 0 & . & . & . & \frac{-h_i}{h_{i-1}} & \frac{h_i}{h_{i-1}} - \frac{h_{i-1}}{h_i} & \frac{h_{i-1}}{h_i} & \vdots & 0 \\ . & . & . & . & . & ... & . & . & . \\ 0 & 0 & . & . & . & ... & . & -1/h_{n-1} & 1/h_{n-1} \end{pmatrix}$$

Cherchons maintenant la matrice S. Pour cela, considérons l'intégrale suivante :

$$\int_{x_1}^{x_n} [s''(t)]^2 dt = \sum_{i=1}^{n-1} \int_{x_i}^{x_{i+1}} [s''(t)]^2 dt$$

En développant et en remplaçant le h (du cas uniforme) par h_i dans cette formule, on obtient une sorte de forme quadratique :

$$\int_{x1}^{xn} [s''(t)]^2 dt = Y''^T S y"$$

Cela nous donne alors la matrice S.

$$S = 1/3 \begin{pmatrix} 2h_1 & \frac{1}{2}h_1 & 0 & 0 & ... & ... & ... & 0 & 0 \\ \frac{1}{2}h_2 & 2h_2 & \frac{1}{2}h_2 & 0 & ... & ... & ... & 0 & 0 \\ \vdots & . & . & . & . & ... & . & . & . \\ \vdots & . & . & . & \frac{1}{2}h_i & 2h_i & \frac{1}{2}h_i & 0 & \vdots \\ . & & & & & & & & \\ . & . & . & . & . & ... & . & . & . \\ 0 & 0 & . & . & . & ... & . & \frac{1}{2}h_{n-1} & 2h_{n-1} \end{pmatrix}$$

Cherchons maintenant les matrices M et N. Pour les obtenir, nous exprimons le vecteur y" en fonction des vecteurs y et y'. Soit s $\in S[\mathrm{x}_1, x_n]$ la spline naturelle correspondant à nos données. Soient $s_i \in \mathbf{R}^3, i = 1, .., n-1$ les polynômes cubiques définissant la spline, avec s_i le polynôme cubique défini entre x_i et x_{i+1}. On applique alors l'interpolation d'Hermite à ces polynômes cubiques avec la condition suivante :

$$y"_i = s"_{i-1}(x_i) = s"_i(x_i), i = 2, ..n-1$$

qui mène à

$$y" = My + Ny'$$

Donc :

$$M = 3 \begin{pmatrix} \frac{1}{h_1^2} & -\frac{1}{h_1^2} - \frac{1}{h_2^2} & \frac{1}{h_2^2} & 0 & 0 & ... & ... & ... & 0 & 0 \\ \vdots & ... & \vdots & . & ... & ... & ... & . & \vdots & \\ \vdots & . & . & . & . & ... & . & . & . & \\ \vdots & . & . & . & \frac{1}{h_{i-1}^2} & -\frac{1}{h_{i-1}^2} - \frac{1}{h_i^2} & \frac{1}{h_i^2} & 0 & \vdots & \\ . & & & & & & & & & \\ . & . & . & . & . & ... & . & . & . & \\ 0 & 0 & . & . & . & ... & \frac{1}{h_{n-2}^2} & -\frac{1}{h_{n-2}^2} - \frac{1}{h_{n-1}^2} & \frac{1}{h_{n-1}^2} & \end{pmatrix}$$

$$N = \begin{pmatrix} \frac{1}{h_1} & \frac{2}{h_1} - \frac{2}{h_2} & -\frac{1}{h_2} & 0 & 0 & ... & ... & ... & 0 & 0 \\ \vdots & ... & \vdots & . & ... & ... & ... & . & \vdots & \\ \vdots & . & . & . & . & ... & . & . & . & \\ \vdots & . & . & . & \frac{1}{h_{i-1}} & \frac{2}{h_{i-1}} - \frac{2}{h_i} & -\frac{1}{h_i} & 0 & \vdots & \\ . & & & & & & & & & \\ . & . & . & . & . & ... & . & . & . & \\ 0 & 0 & . & . & . & ... & \frac{1}{h_{n-2}} & \frac{2}{h_{n-2}} - \frac{2}{h_{n-1}} & -\frac{1}{h_{n-1}} & \end{pmatrix}$$

En appliquant la même méthode que dans le cas uniforme, on construit les matrices H_{03} et H_{12}. Les notations utilisées sont celles du cours de CAO, identiques à celles du cas uniforme.

$$H_{03} = \begin{pmatrix} H_0(t_1^1) & H_3(t_1^1) & & & \\ \vdots & \vdots & & & \\ H_0(t_{k_1}^1) & H_3(t_k 1^1) & & & \\ & H_0(t_{k_1+1}^2) & H_3(t_{k_1+1}^2) & & \\ & \vdots & \vdots & & \\ & H_0(t_{k_2}^2) & H_3(t_{k_2}^2) & & \\ & \ddots & \ddots & & \\ & & & H_0(t_{k_{n-2}+1}^{n-1}) & H_3(t_{k_{n-2}+1}^{n-1}) \\ & & & \vdots & \vdots \\ & & & H_0(t_N^{n-1}) & H_3(t_N^{n-1}) \end{pmatrix}$$

$$H_{12} = \begin{pmatrix} h_1 H_1(t_1^1) & h_1 H_2(t_1^1) & & & \\ \vdots & \vdots & & & \\ h_1 H_1(t_{k_1}^1) & h_1 H_2(t_k 1^1) & & & \\ & h_2 H_1(t_{k_1+1}^2) & h_2 H_2(t_{k_1+1}^2) & & \\ & \vdots & \vdots & & \\ & h_2 H_1(t_{k_2}^2) & h_2 H_2(t_{k_2}^2) & & \\ & \ddots & \ddots & & \\ & & & h_{n-1} H_1(t_{k_{n-2}+1}^{n-1}) & h_{n-1} H_2(t_{k_{n-2}+1}^{n-1}) \\ & & & \vdots & \vdots \\ & & & h_{n-1} H_1(t_N^{n-1}) & h_{n-1} H_2(t_N^{n-1}) \end{pmatrix}$$

Les tests ont été réalisés sur des cas précis et sur un jeu de cas aléatoires.

- Cas d'une distribution uniforme : on trouve le même résultat qu'avec les matrices du cas uniforme. (Figure 4)
- Cas d'une distribution de Chebichev (plus d'explications dans le cours de CAO [1]) (Figure 5)
- Cas aléatoires : on remarque que si la distribution est trop chaotique, alors l'interpolation ne sera plus forcément optimisée. Cela dépend beaucoup des données. (Figures 6 et 7)

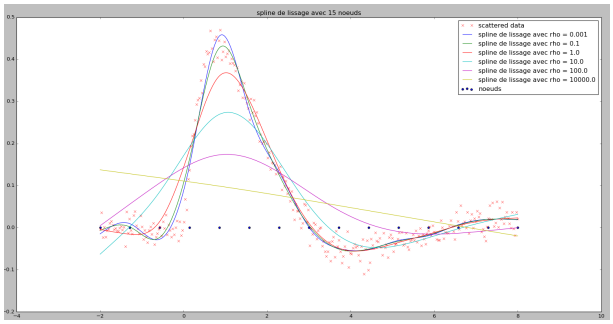


FIGURE 4 – Spline de lissage avec une répartition uniforme

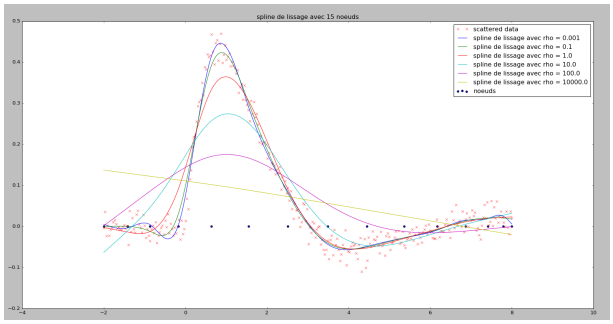


FIGURE 5 – Spline de lissage avec une répartition de Chebichev

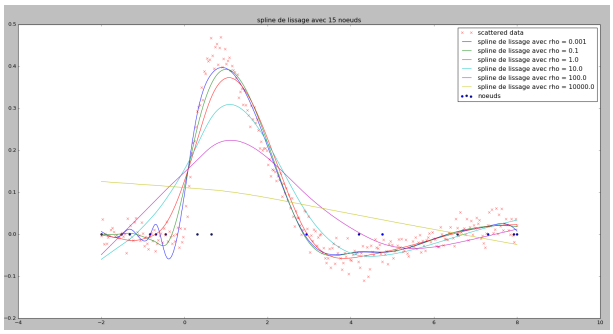


FIGURE 6 – Spline de lissage avec une répartition aléatoire (1)

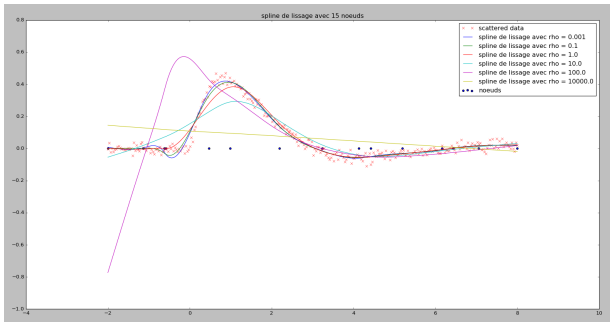


FIGURE 7 – Spline de lissage avec une autre répartition aléatoire (2)

AJOUTER ICI UNE REPARTITION ALEATOIRE TRES FOIREUSE (ne pas oublier la référence)

1.2.3 Choix du paramètre de lissage

Comme vu précédemment, il existe un paramètre de lissage ρ. Ce paramètre peut être vu comme un compromis entre la fidélité de la courbe aux données et la robustesse de l'interpolation. Il exerce une forte influence sur l'estimation, le lissage résultant. Un de nos objectifs porte sur le calcul automatique de ce paramètre par l'intermédiaire de notre application écrite en python.

Après de nombreuses recherches sur une telle optimisation, nous avons conservé la méthode d'optimisation utilisée pour l'estimation par noyau (KDE),problème fondamental de lissage des données, car plus proche de notre projet.

L'estimation par noyau cherche la fonction de densité d'une variable aléatoire (voir une explication détaillée de cette méthode en source [2]).

Pour comprendre, voyons l'équation suivante :

$$\widehat{f}_\rho(x) = \frac{1}{n}\sum_{i=1}^{n} K_\rho(x - x_i) = \frac{1}{n\rho}\sum_{i=1}^{n} K\Big(\frac{x - x_i}{\rho}\Big)$$

On a :

— Un vecteur de coordonnées x, de taille n, considéré comme un échantillon tiré dune distribution dont la densité f est inconnue.

— Un noyau K correspondant à la fonction de densité standard.

— Et enfin ρ le paramètre de lissage (plus communément appelé "bandwidth").

C'est ce paramètre que nous cherchons. Il peut évidemment être choisi de manière subjective : par exemple en précisant sa valeur en fonction de l'expérience précédente. Cependant, une façon plus fiable et objective d'obtenir une valeur pour ce paramètre inconnu est de l'estimer à partir des données de l'échantillon.

On peut illustrer l'importance de son optimisation avec la figure suivante :

En gris, nous avons la densité réelle (qui s'apparente avec celle d'une densité normale dont la moyenne est 0 et la variance est 1). Pour le reste, nous avons 3 estimations de densité :

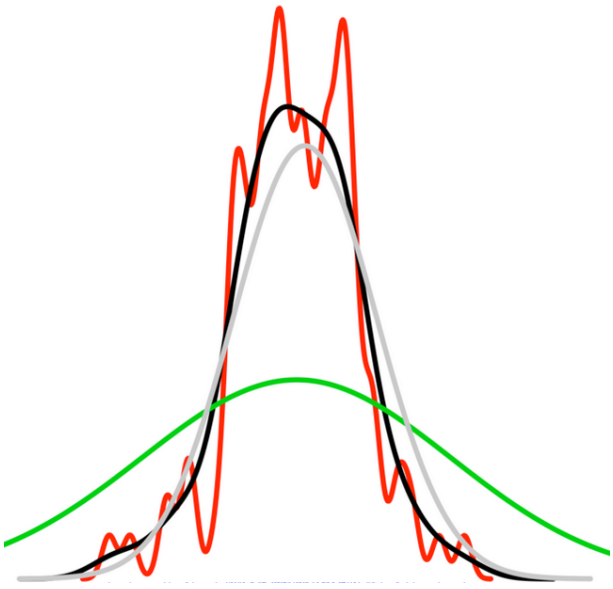


FIGURE 8 – Illustration de différents paramètres de lissage

— Celle en rouge possède un paramètre de lissage h proche de 0 et donc essaye de passer par toutes les données de l'échantillon, mais cela induit qu'il considère toutes les fausses données.

— Celle en vert obscurcit une grande partie de l'information avec un h grand

— Tandis que celle en noir est considérée comme lissée de manière optimale car son estimation est proche de celle réelle.

La méthode nous apprends qu'il y a plusieurs façons :

— La première, la plus générale, est de minimiser l'équation suivante, appelée MISE (mean integrated squared error) [3] :

$$\|\widehat{f}_\rho(x) - f\|_2^2 = \int (\widehat{f}_\rho(x) - f(x))^2 dx$$

— La seconde est de supposer que l'échantillon est distribué selon une loi donnée. Comme dans nos cas, selon la loi normale. Alors d'après nos sources, on pourrait prendre :

$$\rho = 1,06\,\widehat{\sigma}\,n^{-1/5}$$

Après plusieurs tests, nous avons remarqué que nos paramètres de lissage était très similaire sur les deux façons de le calculer (différence des deux inférieur à 10^{-3}). Nous avons donc décidé de garder le premier choix, plus général.

Notre méthode recherche la minimisation des moindres carrés pondérés de façon non linéaire. Il est par conséquent difficile d'avoir un bon jugement sur le fonctionnement de notre méthode. En effet, sur les exemples 9 et 10 ci-dessous, le choix du paramètre du lissage semble logique. Cependant, pour l' exemple 8, la valeur trouvée n'est pas celle qu'on pourrait attendre.

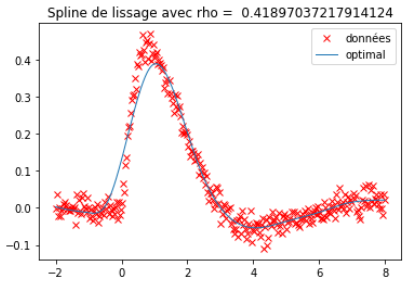


FIGURE 9 – Spline de lissage quelconque

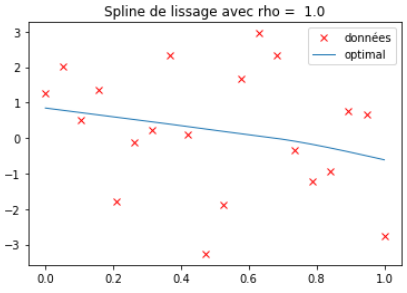


FIGURE 10 – Nuages de points comportant beaucoup de bruit ($\rho = 1$)

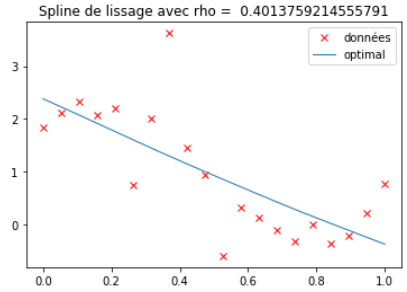


FIGURE 11 – Cas semblant non optimal

Nous avons tenté de trouver l'origine des possibles mauvaises estimations. La plus plausible serait l'intervalle dans lequel l'optimalité du paramètre de lissage est calculé : Dans notre cours de CAO [1], le paramètre de lissage est défini sur R_+. Or, l'estimation par noyau, pourtant consciente que ρ peut avoir une valeur tendant vers l'infini, l'optimalité est calculée seulement sur l'intervalle]0,1], faute de complexité de temps. Nous avons effectué d'autres recherches pour palier à ce problème.
D'autres moyens pour calculer ce paramètre existent. Mais soit le problème d'intervalle est le même (comme par exemple les méthodes de lissage exponentiel de Holt, consistant en des moyennes pondérées des observations passées [4]), soit la difficulté de son implémentation, car portant sur un problème de minimisation des moindres carrés non linéaire, est telle qu'elle n'a pas abouti en temps et en heure sur notre application python (voir pour illustration ce travail de thèse de Khadijetou El Heda portant sur cette optimisation [5]).

Nous resterons donc sur cette méthode d'optimisation provenant estimation par noyau (KDE). Ce choix sera proposé à l'utilisateur un paramètre de lissage basé sur des fondements mathématiques, avant de lui laisser le choix de conserver ce paramètre ou d'en en saisir un autre.

Nous savons donc désormais créer une spline de lissage de façon automatique, cependant celle-ci tient compte de toutes les données sans distinctions. La suite de ce compte rendu va illustrer les techniques d'identification des valeurs aberrantes, d'attribution de poids faibles, et de suppression, et va présenter des algorithmes d'interpolation robustes.

2 Données aberrantes

Il existe plusieurs méthodes différentes pour interpoler des données malgré des points aberrants. La méthode la plus intuitive est de tracer la spline, retirer les points trop éloignés de

celle-ci et recommencer, jusqu'à ce que tous les points soient proches de la spline construite. En faisant des recherches, nous avons découvert deux autres approches : une approche en trois étapes (détection, traitement, construction de la spline), et une approche robuste. Différentes possibilités vont être évoquées dans ce rapport pour chacune de ces approches, après la partie décrivant la méthode intuitive.

2.1 Méthode intuitive

Dans un premier temps, nous avons créé un algorithme qui, à partir d'un point et d'une courbe, renvoie la distance (euclidienne) du point à la courbe.

Ensuite, nous avons écrit et implémenté l'algorithme de comparaison de l'erreur au seuil d'erreur, donné en paramètre, qui permet de déterminer les points aberrants dans les données fournies, par rapport à une spline donnée.

Cela nous a alors permis de créer l'algorithme complet qui calcule une spline, trouve le point le plus éloigné de la courbe et le supprime des données si sa distance à la courbe est supérieure au seuil (ce qui veut dire qu'il est aberrant). On répète ces opérations jusqu'à ce que le point le plus éloigné ne soit plus détecté comme aberrant. La spline est recalculée après chaque suppression de point, car un point aberrant attirait la courbe vers lui inutilement.

Le seuil d'erreur à partir duquel une donnée est décrétée aberrante est très important : il doit être choisi avec soin en fonction des données, et conditionne la précision (et même la réussite) de cette méthode. Plus ce seuil est petit, plus le nombre de points considérés comme aberrants augmente, ce qui peut poser des problèmes de complexité, ou encore engendrer un résultat faux en éliminant des données nécessaires, qui ne sont pas aberrantes.

2.2 Traitement avant la création de la spline

Voici les étapes des algorithmes qui traitent les points aberrants avant de créer la spline :

1. Détection des points aberrants
2. Traitement des points aberrants
3. Calcul et représentation graphique de la spline

2.2.1 Gestion des intervalles locaux

Un problème s'est posé alors que l'on n'y avait pas pensé durant la planification des tâches : beaucoup de méthodes de détection supposent que la répartition des points suit une loi normale. Cela veut dire que les points doivent être proches (en ordonnées) les uns des autres. Pour cela, on doit découper les données à interpoler en plusieurs intervalles qui contiennent des points ayant presque la même valeurs (et éventuellement quelques points aberrants).

En effet, si par exemple on prend la fonction identité discrétisée sur [0,10] avec un point aberrant (1,10), ce point ne sera pas détecté si l'on considère tous les points en même temps, car si celui-ci est décrété aberrant (car il se situe trop loin de la moyenne par exemple), alors (10,10) sera aussi considéré comme aberrant bien que ce ne soit pas le cas.

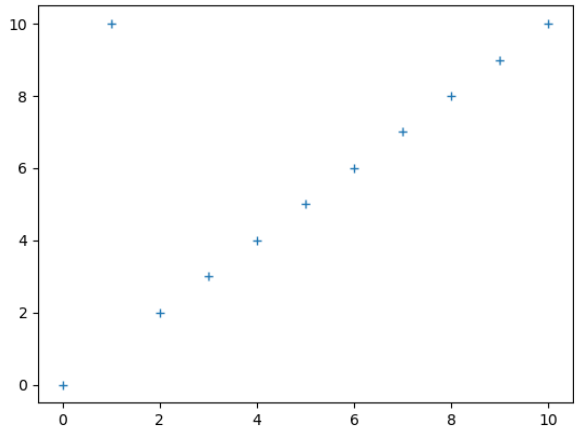


FIGURE 12 – Illustration de l'exemple évoqué

Il faut donc trouver un moyen de séparer les points en groupes de points pas trop éloignés (excepté éventuellement les aberrants) afin de les étudier groupe par groupe.

Pour adapter une série de données en plusieurs petites séries qui semblent suivre une loi normale, nous avons développé deux méthodes. La première, intuitive, ne nous donnait pas satisfaction. C'est pourquoi nous avons cherché à penser d'une manière différente pour développer une seconde méthode, qui n'a rien donc rien à voir avec la première.

2.2.1.1 Méthodes de création d'intervalles

Méthode par pas

Au départ nous avons considéré nos données (x,y) comme des points qui proviennent d'une fonction. Nous avons donc naturellement pensé à créer des intervalles en étudiant la variation de y en fonction de x.

Soit n le nombre de données. Soient $dy_i = |y_{i+1} - y_i|$ et $\Delta y_j = |dy_{j+1} - dy_j|$, avec $0 \leq i < n$, $0 \leq j < n-1$. Cette méthode fonctionne comme suit : on prend un intervalle $[y_d, \ldots, y_f]$ ($d, f \in \mathbf{N}, 0 \leq d \leq f < n$, $d = 0$ et $f = 1$ lors de l'appel initial) et on regarde si $\Delta y_{f-1} < \epsilon$.

- Si $\Delta y_{f-1} < \epsilon$ on rajoute le point suivant à l'intervalle et on teste Δy_f, puis on continue récursivement
- Sinon on ferme l'intervalle courant et on en commence un nouveau qui va contenir initialement deux éléments$[y_f, y_{f+1}]$ (sauf si on est à la fin de l'intervalle).

Pour trouver le ϵ qui convient à chaque série des données, nous avons développé une fonction qui estime ce paramètre automatiquement. Nous commençons par estimer le paramètre en fonction des données, puis nous utilisons la fonction de création d'intervalles à partir du ϵ trouvé, passés en argument.

Méthode par densité

Cette méthode, contrairement à la première, considère les points comme des boules placées sur une barre, sans considérer les ordonnées des données. Nous avons donc pensé à déterminer nos intervalles en fonction de la densité de ces boules sur la barre, c'est à dire en fonction de la densité des abscisses des points sur chaque intervalle.

Pour commencer nous avons considéré notre problème comme un problème de maximisation, c'est à dire comme la recherche d'une méthode qui nous permet de trouver des intervalles dont la densité est maximale.

Pour trouver un résultat optimal, nous avons essayé de modéliser le problème mathématiquement :
soient (x,y) un ensemble de points et ds(d,f) la densité de l'ensemble des points $\{(x_d, y_d), \ldots, (x_f, y_f)\}$sur l'intervalle $[x_d, x_f]$

$$S(d, f) = max(ds(d, f), max_{j=d+1,\ldots,f-1}(ds(d, j) + ds(j, f)))$$

S(d, f) est la somme des densités maximales de tous les intervalles . A partir de cette formule, nous avons développé une méthode qui renvoie les intervalles de densité maximale.
Cette méthode est basée sur trois fonctions :

- densite**(x,d,f)** : cette fonction renvoie la densité de l'ensemble des points $\{(x_d, y_d), \ldots, (x_f, y_f)\}$ sur l'intervalle $[x_d, x_f]$
- ind-int**(x,d)** :cette fonction renvoie l'indice fin de l'intervalle de densité maximale qui commence par l'élément d'indice d .
- ind-densite**(x)** : cette fonction renvoie une liste de tous les intervalles de densité maximale

2.2.1.2 Regroupement d'intervalles

Les intervalles obtenus avec les méthodes précédentes donnent des fois de petits intervalles, sur lesquels certaines méthodes fonctionnent moins bien (en particulier les méthodes interquartile et la méthode des k voisins, qui nécessitent un nombre minimum de données pour renvoyer un résultat correct). Nous avons donc implémenté une fonction qui regroupe tous les intervalles de taille inférieure à un paramètre avec leurs voisins jusqu'à ce que la taille de tous les intervalles soit supérieure ou égale à ce paramètre.

2.2.2 Méthodes de détection des points aberrants

Dans cette partie, nous allons présenter plusieurs méthodes de détection de points aberrants. Elles nécessitent, pour certaines, des paramètres. Cela peut être un coefficient, un taux d'erreur... Ceux-ci sont à adapter manuellement à chaque exemple. Les estimer automatiquement serait possible, mais nous ne l'avons pas fait, par manque de temps.

2.2.2.1 Méthode inter-quartiles

Cette méthode, comme son nom l'indique, détecte les points aberrants en utilisant le 1er et le 3e quartile. Cette méthode est aussi nommée méthode de la boîte à moustaches, ou encore méthode de la boîte de Tukey.

L'algorithme de cette méthode est implémenté en deux fonctions. La première fonction calcule dans un premier temps les premiers et troisièmes quartiles afin de récupérer l'écart inter-quartiles. Ensuite, elle construit et renvoie un intervalle de confiance , défini de la manière suivante, avec $Q1$ et $Q3$ les premier et troisième quartiles, é$cart_interquartile$ la distance entre $Q1$ et $Q3$ (é$cart_interquartile = Q3 - Q1$), et $coeff$ un certain coefficient : $[Q1 - coeff * (Q3 - Q1), Q3 + coeff * (Q3 - Q1)]$.

Il faut fixer $coeff$ correctement : s'il est trop grand, nous ne détecterons pas les points aberrants tandis que s'il est trop petit, des points seront déclarés aberrants alors qu'ils ne le sont pas. Beaucoup de sources (en particulier [6] et [7]) conseillent de fixer $coeff$ à 1.5, ce que nous faisons.

Le point est considéré comme aberrant si et seulement si il se situe dans cet intervalle.

2.2.2.2 Test Tau de Thompson

Le test Tau de Thompson est un test statistique qui détecte les valeurs aberrantes parmi une série de valeurs. Ce test est considéré comme un des meilleurs critères de détection des valeurs aberrantes parce qu'il les détecte statistiquement en tenant compte de l'écart type et de la moyenne.

Pour ce test on définit deux quantités : $\delta_x = \frac{x-\bar{x}}{s}$ et seuil $= \frac{t_{\alpha/2}*(n-1)}{\sqrt{n}*\sqrt{n-2+t^2_{\alpha/2}}}$

- $\bar{x}$ est la moyenne de la série de valeurs
- s est l'écart type de la série de valeurs
- x est la valeur testé e
- α est un paramètre que l'on fixe
- $t_{\alpha/2}$ est la valeur critique provenant de la loi de Student avec une probabilité égale à $\alpha/2$ et n-1 degré de liberté
- n est le nombre de valeurs

Une valeur x est dite aberrante uniquement si δ_x est supérieur au seuil ou inférieur à l'opposé du seuil (-seuil).

2.2.2.3 Test de Chauvenet

Cette méthode permet de détecter les données aberrantes parmi une série de données. Ce test considère que les données étudiées suivent une loi normale de moyenne égale à la moyenne empirique et de variance égale à la variance empirique.

Le principe de cette méthode est simple : pour chaque donnée testée x_t, on calcule la probabilité d'avoir une donnée qui s'écarte de plus de $|x_t - \bar{x}|$ de la moyenne et on la multiplie par la taille de l'échantillon n.
$N = n * P(|X - \bar{x}| > |x_t - \bar{x}|)$ avec $\bar{x}$ la moyenne empirique
Si N est inférieur à tau, on considère que la donnée testée est aberrante sinon on admet qu'elle ne l'est pas. Tau est un paramètre de la méthode.

2.2.2.4 Test de Grubbs

Cette méthode a été inventée par Frank E. Grubbs en 1969. Pour ce test, on n'étudie que la valeur extrême (celle dont la valeur absolue de l'écart à la moyenne est la plus grande). Si plusieurs existent, on peut itérer ce test plusieurs fois tant que l'on trouve des points aberrants, en retirant le point aberrant trouvé.

L'algorithme fonctionne comme suit : la valeur extrême est récupérée, puis l'on compare ensuite $\frac{v_{extrême}-moyenne}{écart\_type}$ avec le seuil critique donné par le test de Grubbs : $G_{crit} = \frac{n-1}{\sqrt{n}}\sqrt{\frac{t^2_{\frac{\alpha}{n},n-2}}{n-2+t^2_{\frac{\alpha}{n},n-2}}}$, avec n le nombre de données, $t_{a,b}$ le résultat de la fonction quantile de Student avec un seuil de confiance a et b degrés de liberté, et α l'erreur que l'on accepte. Si la première valeur est plus grande que le seuil, alors la méthode de Grubbs considère que le point extrême est aberrant.

α est un paramètre que nous décidons de passer à la fonction. Plus celui-ci est faible, plus la chance que les points détectés comme aberrants le soient réellement (mais dans ce cas, peu sont détectés, certains points pourtant aberrants peuvent ne pas être détectés)

Pour plus de détails sur cette méthode, se référer aux sources [8] et [9].

2.2.2.5 Méthode de la déviation extrême de Student

Cette méthode a été inventée par Bernard Rosner en 1983. Ce test est une généralisation du test de Grubbs. En anglais, ce test appelé "extreme studentized deviate" est abbrégé ESD. L'algorithme suit les étapes suivantes :
- Récupération des valeurs extrêmes (même définition que dans le paragraphe précédent)
- Comparaison des valeurs extrêmes normalisées avec le seuil critique, qui dépend du nombre de valeurs extrêmes déjà enlevées. Valeur normalisée : $\frac{v_{extreme}-moyenne}{ecart-type}$. Seuil critique, avec i le nombre de valeurs déjà enlevées : $\frac{(n-i-1)*t_{\frac{1-\alpha}{2(n-i)},n-i-2}}{\sqrt{(n-i)*(n-i-2+t^2_{\frac{1-\alpha}{2(n-i)},n-i-2})}}$
- On ne compare pas les valeurs suivantes si un point est décrété comme non aberrant : en effet, on traite les valeurs dans l'ordre de leur "extrêmitude".

Plus d'informations sur cette méthode sont trouvables à la source [10].

2.2.2.6 Méthode des k plus proches voisins

Cette méthode est aussi appelée KNN ou k-NN, pour k nearest neighbors. Elle porte bien son nom : chaque point est comparé à ses k plus proches voisins afin de savoir s'il est aberrant ou non.

Plus précisément, cette méthode cherche, pour chaque donnée, ses k plus proches voisins. L'éloignement est mesuré selon la distance euclidienne, mais on pourrait choisir une autre distance facilement. On ordonne ensuite les données selon la moyenne de la distance de ses k plus proches voisins. Les données qui obtiennent alors les plus grandes moyennes sont des données aberrantes (car leurs voisins sont trop éloignés). Le pourcentage de données aberrantes est un paramètre de la méthode.

Cette méthode a cependant un problème de complexité : la recherche des k voisins nécessite, pour chaque donnée, de parcourir toutes les autres. Cela rend les calculs longs si les données sont nombreuses.

Plus k est petit, plus la méthode semble précise. Cependant, si k est trop petit, des données seront détectées alors qu'elles ne sont pas aberrantes. sur les valeurs aberrantes et donne un meilleur nuage de données. Un autre argument très important est la proportion de paramètres à détecter : il faut l'adapter manuellement aux données.

Les sources [11], [12] et [13] donnent plus d'informations sur cette méthode et ses limites.

2.2.3 Détection et traitement : winsorization

La Winsorization est une manière de traiter les points extrêmes, qui sont en général les points aberrants. Cette méthode effectue les étapes 1 et 2 de la création de la spline avec traitement préalable, ce qui veut dire qu'elle détecte et traite les points aberrants, mais ne construit pas la spline. Il suffit ensuite de construire la spline de lissage associée aux données modifiées par cette méthode, pour avoir l'interpolation des données malgré les points aberrants.

Cet algorithme est plutôt intuitif : il récupère le pourcentage de valeurs données comme aberrantes et modifie les valeurs extrêmes. La modification est équitablement répartie entre les valeurs trop grandes et celles trop petites. La valeur affectée à ces points est la plus proche (la plus grande pour les valeurs trop grandes, la plus petite pour celles trop petites) qui n'est pas aberrante.

Cet algorithme fonctionne également si le pourcentage donné est (un peu) trop grand. En effet, si les valeurs ne sont pas aberrantes, elles vont être proches des autres, et leur valeur ne sera donc que très peu modifiée.

2.2.4 Traitement des points détectés

Dans cette partie, nous allons évoquer les différents traitement possibles des points aberrants avant l'interpolation des données.

Deux algorithmes ont été implémentés pour le traitement des points (chacun pouvant être utilisé pour chaque méthode de traitement). AMELYS, EST CE QUE LES DEUX SONT ENCORE UTILISES OU JUSTE UN SEUL ? MODIFIER SI C'EST DEVENU FAUX ! Le premier algorithme prend en argument traite toutes les données en une fois, tandis que le deuxième ne traite que le point dont l'indice est passé en paramètre.

2.2.4.1 Suppression

La méthode la plus simple est de simplement supprimer les points aberrants des données utilisées pour le calcul de la spline. Cet algorithme est indépendant de la méthode de détection de points aberrants utilisée.

2.2.4.2 Méthode inspirée de la Winsorization

La Winsorization, que nous avons déjà évoquée au paragraphe 2.1.2.1, nous a donné l'idée d'un autre traitement de valeurs aberrantes : les remplacer par la valeur non aberrante la plus proche. Ce n'est pas la Winsorization, car on ne supprime pas équitablement les valeurs trop grandes et celles trop petites.

2.2.4.3 Attribution de poids

Chaque méthode de détection utilisée nous fournit un vecteur de poids qui affecte un poids aux points détectés comme aberrants strictement inférieur à 1, et un poids fort égal à 1 pour ceux qui n'ont pas été détectés comme tels.

Il n'existe pas de valeur rigoureuse pour ce poids faible, mais voulant atténuer nos points aberrants au profit des autres et non pas les supprimer, nous leur attribuerons un poids égal à 1/3. Nous verrons dans la partie suivante, consacrée à la construction d'une spline, la façon dont nous avons utilisé ce vecteur.

2.2.5 Construction de la spline

La construction de la spline diffère en fonction des cas : si les données ont été pondérées, nous n'allons pas construire la spline de la même façon que si elles ne l'ont pas été.

2.2.5.1 Données non pondérées

L'utilisateur peut choisir de construire une spline dont ses points aberrants seront supprimés, où de les traiter à la manière Winsorizing. Cela veut dire que tous les points restant (incluant ceux modifiés à la manière de Winsorizing) seront interpolés pour créer la spline. Pour plus d'information, se référer aux explications sur la construction d'une spline de lissage (Section 1.2 du rapport).

2.2.5.2 Pondérée

Un traitement possible des valeurs aberrantes est de leur affecter un poids. On souhaite dans ce cas que la spline construite tienne compte de notre vecteur de poids, c'est à dire qu'elle soit moins attirée par les données aberrantes.

L'idée que nous avons eu pour utiliser notre vecteur dans la construction de la spline, c'est d'intégrer notre vecteur de poids dans une méthode faisant elle-même intervenir des poids. C'est là qu'intervient la méthode de régression linéaire pondérée LOESS, une méthode robuste réalisant à la base un travail similaire à celui de nos méthodes construisant nos splines, mais en faisant intervenir des poids de façon locale. Nous avons ici adapté cette méthode afin qu'elle considère notre vecteur de poids lors l'interpolation de la spline (expliquée de façon détaillée en 2.3.2). Cela veut dire que l'impact des points aberrants sur les points considérés comme "proches" d'eux sera davantage réduit.

Cet exemple illustre la construction de la spline à partir de données pondérées :

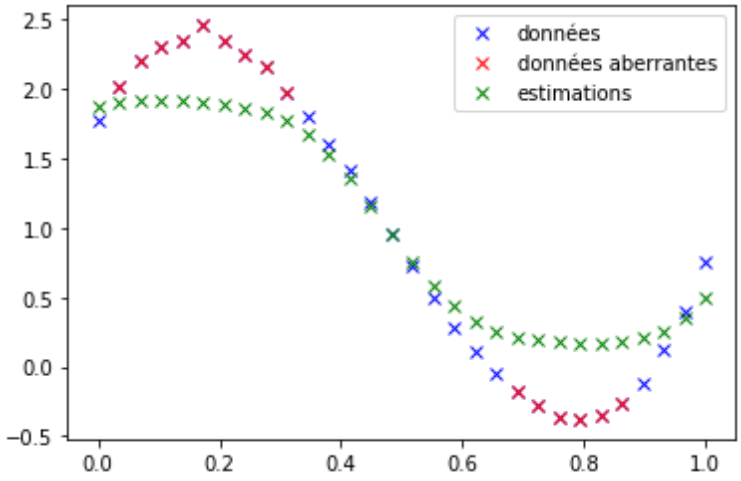


FIGURE 13 – Exemple de construction

Les données en vert sont les données estimées par notre méthode LOESS modifiée avec un ρ optimal de 0.05. Ce sont ces points estimés qui vont servir à créer une la nouvelle spline, moins influencée par les valeurs détectées comme aberrantes par nos méthodes.

Nous avons donc une méthode utilisant notre vecteur de poids donné par nos méthodes de détection et rendant les splines moins influencées par les points aberrants, ce que nous voulions.

2.3 Interpolation robuste

Nous avons également trouvé lors de nos recherches des algorithmes robustes, c'est à dire des algorithmes dont le résultat n'est pas réellement influencé par les points aberrants. Ces méthodes calculent la spline tout en détectant les points aberrants au fur et à mesure.

2.3.1 Algorithme de RANSAC

RANSAC est l'abréviation de RANdom SAmple Consensus. Cet algorithme est non déterministe, cela veut dire qu'il peut renvoyer des résultats différents pour une même entrée (mêmes données et paramètres) à cause de l'utilisation du hasard.
Cet algorithme va calculer un bon nombre de splines et va choisir celle qui correspond aux données mais qui engendre le moins d'erreur possible entre la spline et les données décrétées comme non aberrantes. Voilà les étapes :

1. L'algorithme choisit un certain nombre de points (distincts) aléatoires
2. Il calcule la spline cubique passant par ces points
3. Il mesure de la distance entre la courbe et chaque point afin de récupérer les points non aberrants d'après la spline créée
4. Si les points non aberrants trouvés lors de l'étape précédente sont peu nombreux, au moins un point utilisé pour créer la spline a de grande chances d'être aberrant. L'algorithme passe donc à une nouvelle spline, en retournant à l'étape 1.
5. Si la spline proche de beaucoup de points, c'est qu'à priori elle interpole la majorité des données. Les autres points sont décrétés comme aberrantes. L'algorithme calcule la spline de lissage correspondant aux données considérées comme non aberrantes.
6. La distance totale des points (non aberrants) par rapport à la courbe est calculée, c'est l'erreur de la spline de lissage étudiée. On sauvegarde cette spline si et seulement si elle est meilleur que la précédente (c'est à dire si cette erreur est plus faible), ou si c'est la première qui correspond aux données. Une fois qu'un certain nombre de splines a été testé, c'est la dernière sauvegardée qui est considérée comme l'interpolation des données.

Avec cette version de l'algorithme, beaucoup de paramètres sont nécessaires (en plus des données) et doivent être réglés manuellement, en fonction des données et de ce que l'on souhaite obtenir :

- Le nombre de spline à créer
- L'erreur acceptée entre les points et la spline cubique jusqu'à laquelle les points ne sont pas considérés comme aberrants à l'étape 3
- La fonction de distance à utiliser aux étapes 3 et 6
- Le nombre de points non aberrants à partir duquel on considère que la spline correspond aux données
- Le nombre de points à considérer lors de la construction de la spline de l'étape 2 (c'est à dire le nombre de points tirés à l'étape 1)
- Le paramètre de lissage de la spline à l'étape 5.

Deux paramètres sont cependant estimables au fur et à mesure de l'algorithme :
le nombre de spline à créer (c'est à dire le nombre d'itérations), et le nombre de points minimum à considérer pour avoir un résultat correct. Cela n'est possible qu'à condition de fournir à la place la probabilité d'avoir un résultat correct (cette probabilité doit se trouver dans $]0, 1[$.).
Les calculs, détaillés à la 6ième page du document [14], nous donne le nombre d'itérations à effecteur pour avoir une probabilité $p_{correct}$ d'avoir un résultat correct, avec une proportion ω de données non aberrantes (aussi appelées inlier) : $n_{iter} = \frac{log(1-p_{correct})}{log(1-\omega^n)}$

Les deux versions de l'algorithme évoquées ont été implémentées dans l'application, mais seule la seconde peut être lancée et appliquée sur les données. Une première fonction implémente la première version évoquée, tandis qu'une seconde calcule au fur et à mesure la proportion de données non aberrantes de l'échantillon et actualise le nombre maximum d'itérations grâce au résultat évoqué précédemment. C'est avec cette seconde que les tests et les comparaisons ont été effectués.

Cet algorithme possède plusieurs limites. On va essayer d'en expliquer quelques-unes :

- Il faut, pour l'instant, définir certains paramètres manuellement, en fonction des exemples.
- L'algorithme est non déterministe, ce qui signifie que le résultat peut, avec peu de chances, être totalement faux. De plus il varie en fonction des essais.
- Le paramètre de lissage est le même sur toute la fonction donc si celle-ci est stable à un endroit mais bouge beaucoup ailleurs, on ne peut pas avoir les deux. (test 22)
- Il est impossible (ou très difficile) de savoir si certains points sont aberrants ou pas (exemple num 20) : comment savoir si ce sont les deux/trois points du haut ou du bas qui sont aberrants ? (à gauche)

- Certains points sont décrétés comme aberrants alors que visuellement, ils ne le sont pas. (Exemple num 11)

ENLEVER LES NUMEROS DE TEST, MAIS METTRE UNE IMAGE COMME PREUVE POSSIBLE QUE LORSQUE L'ESTIMATION DU PARAMETRE NE SERA PAS OBLIGATOIREMENT AUTOMATIQUE

2.3.2 Méthode de LOESS

LOESS (ou LOWESS) est une méthode robuste de régression non paramétrique. Elle utilise la régression linéaire des moindres carrés pondérés, en considérant de manière plus importante les données les plus proches de ce point.

L'objectif de cette méthode est d'ajuster $\theta = [\theta_0, \theta_1]$ pour minimiser les moindres carrés pondérés, c'est à dire la quantité suivante :

$$\sum_{i=1}^{m} w_i(y_i - (\theta_0 + \theta_1 x_i))^2 \quad (1)$$

Avec :

- θ_0 *et* θ_1 les paramètres inconnus dont la valeur permet d'effectuer la procédure d'ajustement
- $(\theta_0 + \theta_1 x_i)$ la coordonnée en y prédite par la méthode en fonction de ces paramètres.
- $1 > w_i = \exp\left(-\frac{(x-x_i)^2}{2\rho}\right) > 0$ le poids Gaussien où ρ est considéré dans notre cas comme le paramètre de lissage.

Les poids sont donc donnés en utilisant le calcul des moindres carrés, on a par conséquent :

- Plus de poids à des points près du point cible x
- Moins de poids à des points plus loin de x

Autrement dit, si la différence $|x_i - x|$ est petite, alors le poids w_i est proche de 1. Tandis que dans le cas contraire, si elle est grande, alors w_i est proche de 0. Notre modèle est ajusté une fois la méthode appliquée, pour ne retenir que le point du modèle qui est proche du point cible. La procédure se répète pour chaque point par ordre croissant des abscisses.

Partons de l'expression (1) avec x et y des vecteurs de taille m. Appelons cette expression S en fonction de θ, nous avons :

$$S(\theta) = \sum_{i=1}^{m} w_i \left(y_i - (\theta_0 + \theta_1 x_i)\right)^2$$

$$\frac{\partial S}{\partial \theta_0} = -2 \sum_{i=1}^{m} w_i \left(y_i - (\theta_0 + \theta_1 x_i)\right)$$

$$\frac{\partial S}{\partial \theta_1} = -2 \sum_{i=1}^{m} w_i \left(y_i - (\theta_0 + \theta_1 x_i)\right) x_i$$

Puis :

$$\frac{\partial S}{\partial \theta_0} = \sum_{i=1}^{m} w_i \left(y_i - (\theta_0 + \theta_1 x_i)\right) = 0$$

$$\iff \sum_{i=1}^{m} w_i \theta_0 + \sum_{i=1}^{m} w_i \theta_1 x_i = \sum_{i=1}^{m} w_i y_i \quad \text{Eq. (1)}$$

$$\frac{\partial S}{\partial \theta_1} = \sum_{i=1}^{m} w_i \left(y_i - (\theta_0 + \theta_1 x_i)\right) x_i = 0$$

$$\iff \sum_{i=1}^{m} w_i\theta_0 + \sum_{i=1}^{m} w_i\theta_1 x_i x_i = \sum_{i=1}^{m} w_i y_i x_i \quad \text{Eq. (2)}$$

En écrivant les équations Eq. (1) et Eq. (2) sous forme de matrice $\mathbf{A}\theta = \mathbf{b}$ nous obtenons :

$$\begin{bmatrix} \sum w_i & \sum w_i x_i \\ \sum w_i x_i & \sum w_i x_i x_i \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} \sum w_i y_i \\ \sum w_i y_i x_i \end{bmatrix}$$

$$\iff \mathbf{A}\theta = \mathbf{b}$$

$$\iff \theta = \mathbf{A}^{-1}\mathbf{b}$$

Il nous suffit donc de résoudre cette matrice et de combiner notre vecteur de poids avec celui de cette méthode, soit :

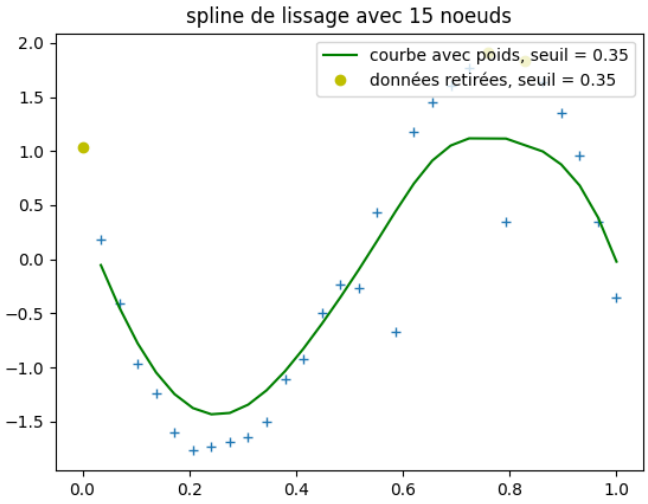
$$1 > w_i = \exp\left(-\frac{(x - x_i)^2}{2\rho}\right) * vpoids_i > 0$$

où *vpoids* est notre vecteur de poids trouvé avec l'aide de nos méthodes de détection et affectant un poids faible aux points aberrants. Ainsi, pour n'importe quel point étudié, les points aberrants ont un impact réduit sur le tracé de la spline.

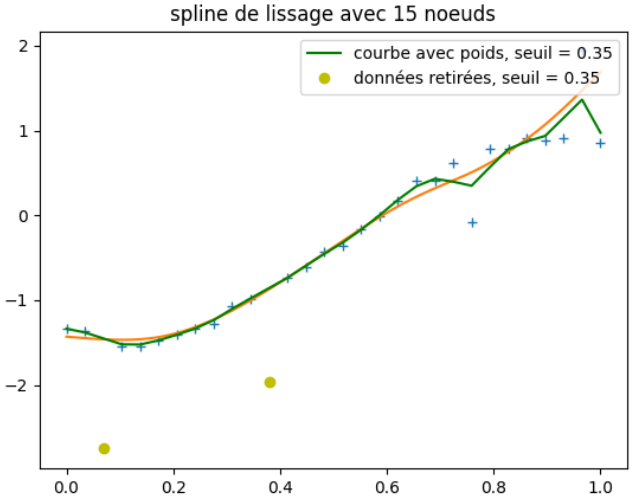
IV
Tests et comparaison des différentes méthodes de gestion de points aberrants

1 Méthode intuitive

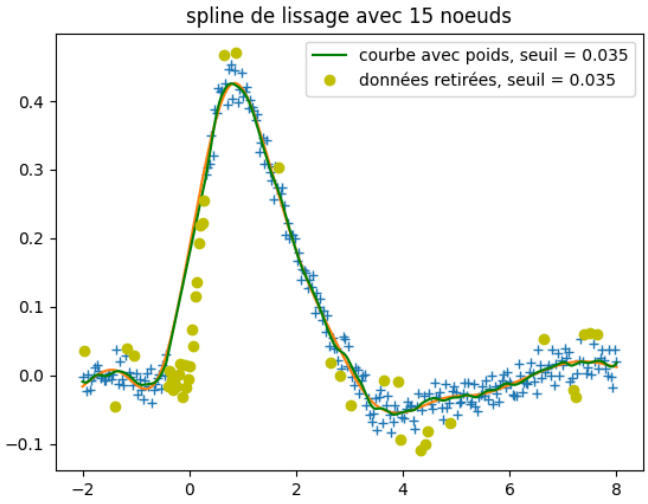
TROUVER AUSSI DES EXEMPLES NE MARCHANT PAS CORRECTEMENT.


spline de lissage avec 15 noeuds
courbe avec poids, seuil = 0.35
données retirées, seuil = 0.35
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.0
0.2
0.4
0.6
0.8
1.0

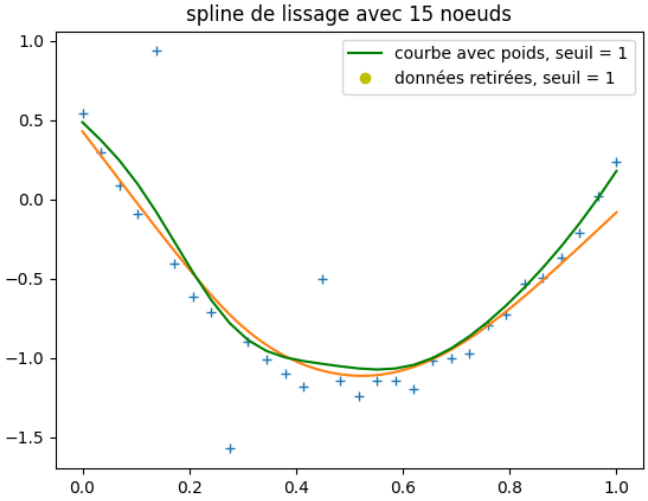
ICI


spline de lissage avec 15 noeuds
courbe avec poids, seuil = 0.35
données retirées, seuil = 0.35
2
1
0
−1
−2
0.0
0.2
0.4
0.6
0.8
1.0

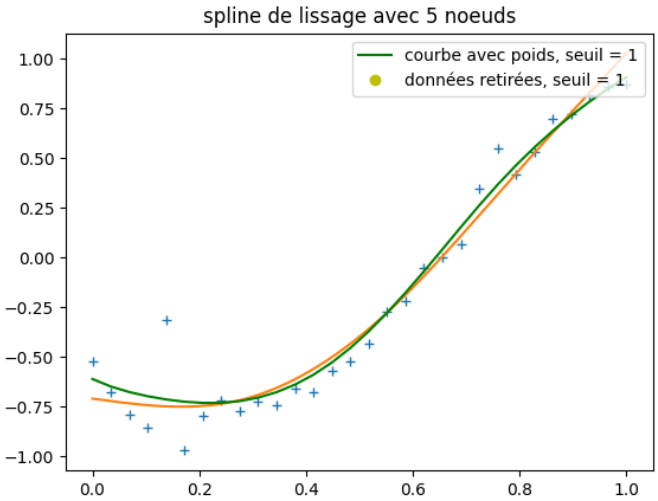
ICI


spline de lissage avec 15 noeuds
courbe avec poids, seuil = 0.035
données retirées, seuil = 0.035
0.4
0.3
0.2
0.1
0.0
−0.1
−2
0
2
4
6
8

ICI


spline de lissage avec 15 noeuds
courbe avec poids, seuil = 1
données retirées, seuil = 1
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.0
0.2
0.4
0.6
0.8
1.0

ICI



ICI

2 Détection, traitement, et interpolation

Après plusieurs tests de chaque méthode de détection des données aberrantes, nous avons constaté deux choses :

- Les paramètres de toutes les méthodes de détection semblent dépendre de la taille de l'échantillon et non pas de la distribution des données
- Ces paramètres semblent se stabiliser lorsqu'on augmente la taille de l'échantillon, autrement dit, ces paramètres convergent vers les valeurs fixées par défaut dans chaque fonction, lorsque la taille de l'échantillon augmente (≥ 100)

Voici ci-dessous le résultat d'une méthode particulière avec différentes tailles. Les données en rouges sont les données conservées tandis que celles en bleu sont considérées comme aberrantes par l'algorithme. La courbe bleue représente l'interpolation attendue.

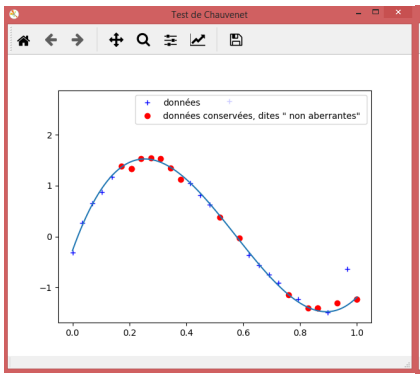


FIGURE 14 – Taille de l'échantillon = 30

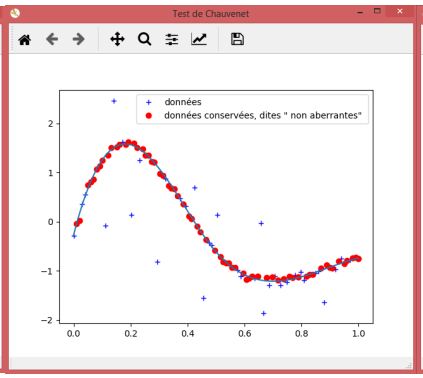


FIGURE 15 – Taille de l'échantillon = 100

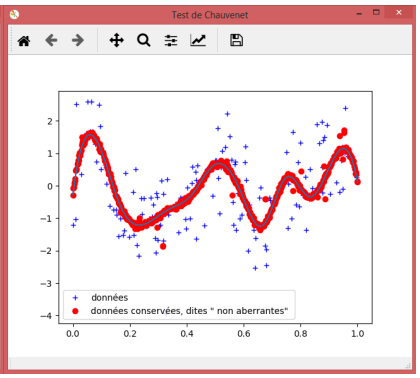


FIGURE 16 – Taille de l'échantillon = 1000

Pour tous les tests qui viennent, nous allons fixer la taille de l'échantillon à 200 pour les signaux stationnaires et à 300 pour les signaux non-stationnaires.

2.1 Création des intervalles

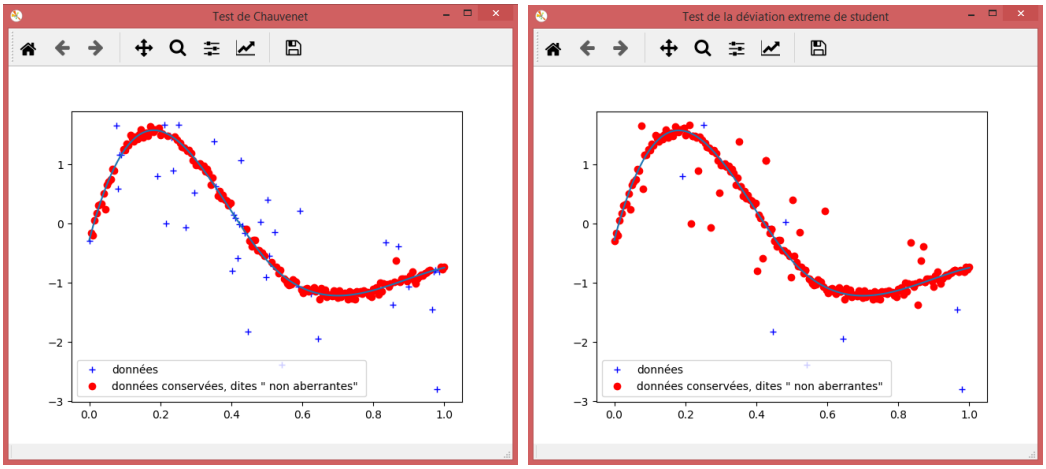
Pour tester ces méthodes, nous avons simplement détecté les points aberrants. Pour reproduire ces tests, il faut choisir l'option ... AMELYS AJOUTER LA BONNE OPTION ICI. En effet, nous avons ici considéré seulement la détection, et donc ignoré les étapes de traitement de points aberrants et la construction de la spline.

2.1.1 Création d'intervalles selon un pas

Cette méthode, comme mentionné dans la partie la concernant, ne fonctionne pas dans tous les cas. Voici ci-dessous quelques cas avec lesquels elle fonctionne et d'autres avec lesquels elle ne fonctionne pas. Dans chaque exemple, la graine a été fixée afin de pouvoir reproduire le même test. La graine (seed) est celle fournie au générateur de signal : cela permet de "fixer" le hasard.

2.1.1.1 Cas stationnaire (seed = 5505361)

Dans le cas d'un signal variant peu, cette méthode est correcte avec certaines méthodes de gestion de données aberrantes (test de Chauvenet, ..., ...), mais pas avec toutes (déviation extrême de Student, ..., ...). (A COMPLETER Béryl)

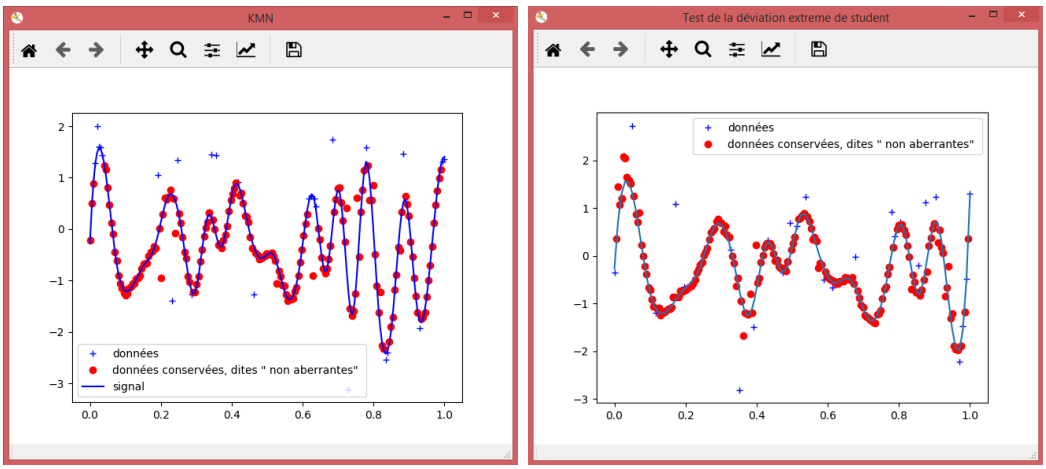


Test de Chauvenet, régularité = 0.9

Déviation extrême de Student, régularité = 0.9

FIGURE 17

Cette méthode de création d'intervalles fonctionne mieux dans le cas de signaux avec de grandes variations comme l'illustrent les exemples suivants.



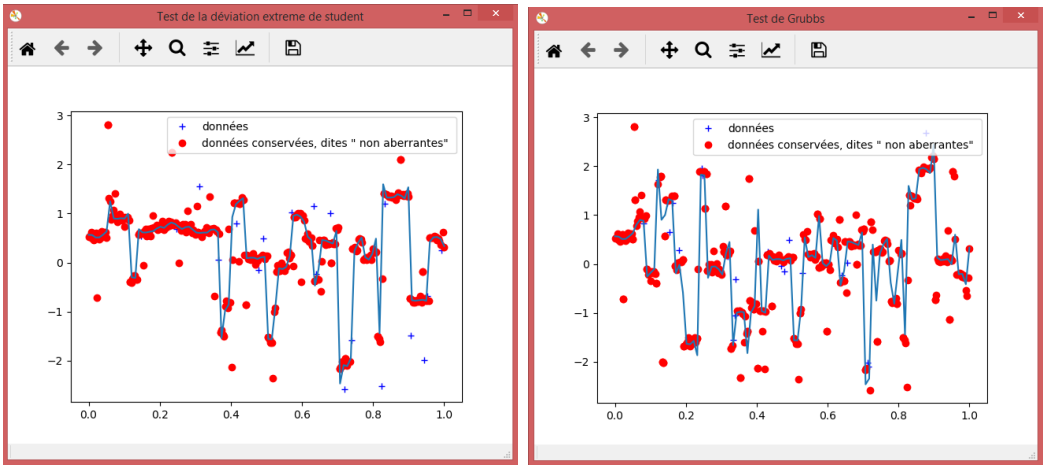
k plus proches voisins,
régularité = 0.65

Déviation extrême de Student,
régularité = 0.7

FIGURE 18

2.1.1.2 Cas non-stationnaire (seed = 32427415)

Les cas qui ne fonctionnent pas dans le cas stationnaire ne fonctionnent pas non plus dans le cas stationnaire, comme le montrent entre autres les figures suivantes.

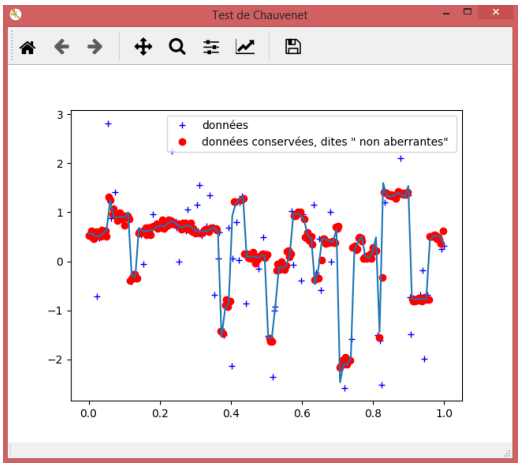


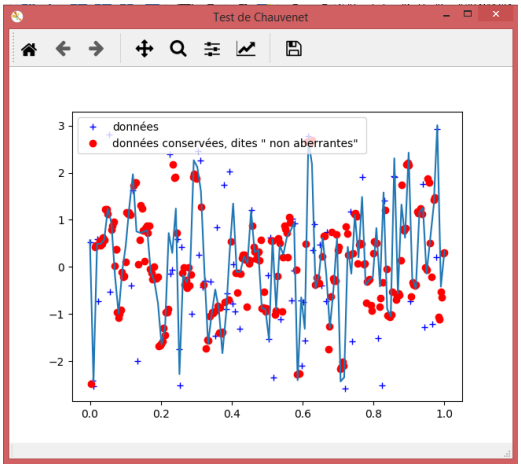
Déviation extrême de Student,
switch-prob = 0.1

Test de Grubbs,
switch-prob = 0.2

FIGURE 19

Dans les cas où la méthode fonctionne sur un signal stationnaire, le résultat pour un signal non stationnaire est satisfaisant lorsqu'on prend de petites valeurs de switch-prob (paramètre lié à l'apparition d'une grande variation à chaque instant, donc lié à la largeur des créneaux) et reste acceptable pour de grandes valeurs, bien que moins précis.





Test de chauvenet,
switch-prob = 0.1

Test de chauvenet,
switch-prob =0.5

FIGURE 20

2.1.2 Création d'intervalles selon la densité

Cette méthode semble renvoyer un résultat correct dans tous les cas étudiés précédemment (stationnaire et non stationnaire, petites et grandes variations) et avec toutes les méthodes de détection de points aberrants étudiées.

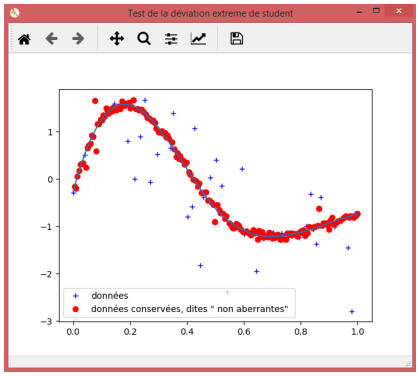


FIGURE 21 –
Déviation extrême de Student,
régularité = 0.9

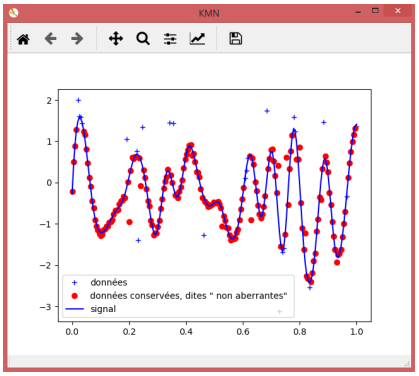


FIGURE 22 –
k plus proches voisins,
régularité = 0.65

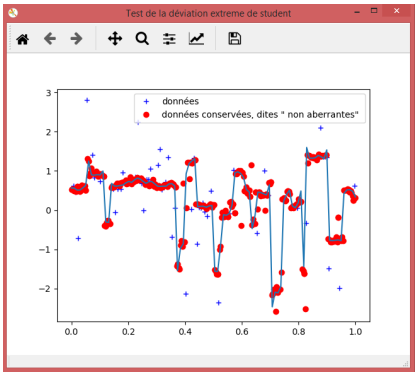


FIGURE 23 –
Déviation extrême de Student,
switch-prob = 0.1

Cette méthode admet cependant la même limite que la précédente, dans le cas de signaux non stationnaires avec de grandes valeurs de switch-prob : elle fonctionne moins bien que ce que l'on pourrait attendre sur certains exemples. Voici ci-dessous deux exemples qui illustrent cette limite. On constate que les résultats obtenus restent pas très éloignés de ceux attendus : ils ne sont pas totalement faux, juste moins précis que ce qu'on pourrait attendre.

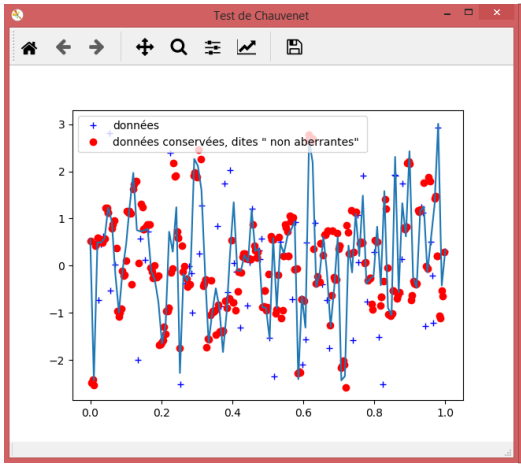


FIGURE 24 –
Test de Chauvenet,
switch-prob = 0.5,seed = 32427415

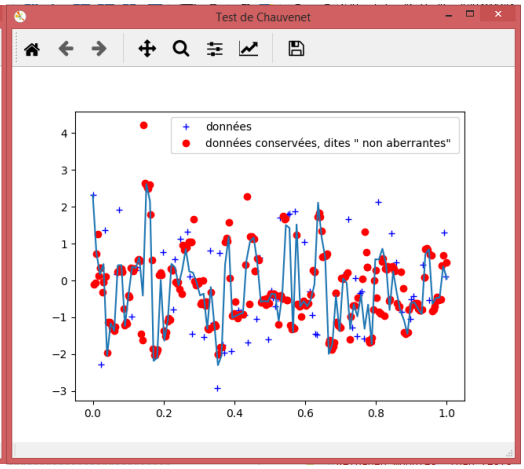


FIGURE 25 –
Test de Chauvenet,
switch-prob = 0.5, seed = 54441996

2.1.3 Regroupement

On a évoqué, lors de l'explication de l'utilité de cette fonction, que deux méthodes de détection de points aberrants nécessitent que les intervalles ne soient pas trop petits afin de fonctionner correctement : les méthodes interquartile et k plus proches voisins. L'exemple ci-dessous montre la détection obtenue avec l'une des deux méthodes, mais c'est similaire pour la seconde. Le paramètre t est le nombre minimal de points par intervalle.

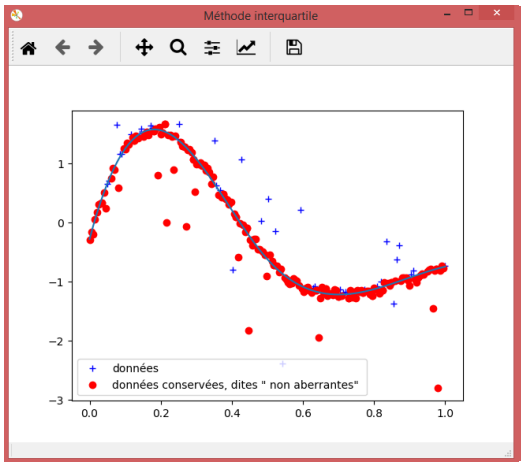


FIGURE 26 – inter-quartile, t=5

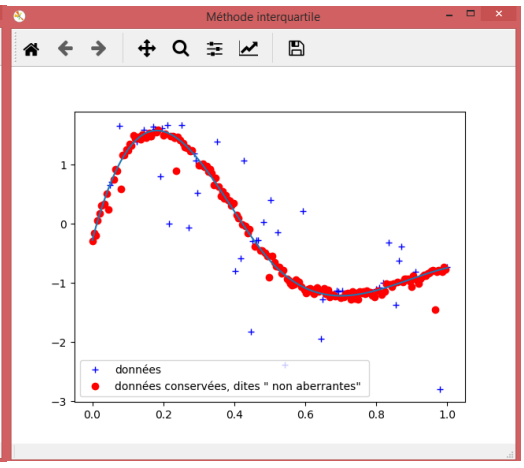


FIGURE 27 – inter-quartile, t=10

Pour fixer le t, nous devons cependant voir s'il n'existe pas une valeur à partir de laquelle les résultats deviennent moins bons, car on retombe sur la même problématique qu'en l'abscence d'intervalles. Voici l'exemple précédent avec 30 points minimum par intervalle, qui est moins bon que celui avec t = 10.

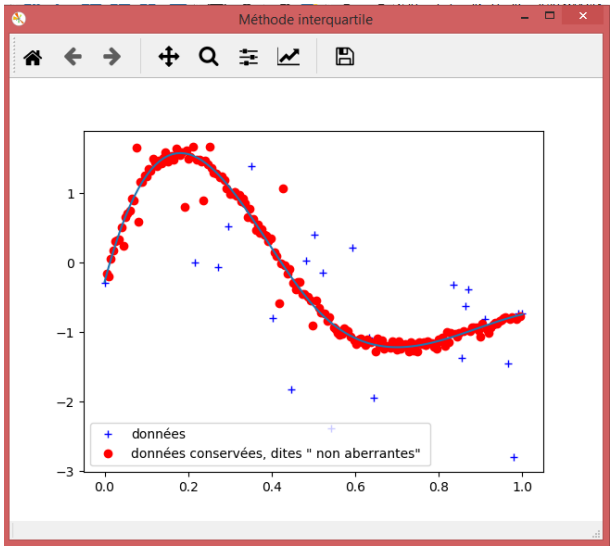


FIGURE 28 – inter-quartile, t=30

Après plusieurs autres tests, nous avons fixé le paramètre t de la fonction regroupe à 10 pour la méthode interquartile et à 30 pour la méthode des k plus proches voisins.

2.2 Détection

Les six méthodes de gestion de points aberrants donnent un résultat plus ou moins similaire, peu importe le signal choisi. On ne peut pas dire qu'une méthode fonctionne mieux que les autres parce qu'on peut toujours améliorer le résultat de chaque méthode en trouvant la valeur exacte de son paramètre pour un l'échantillon de points considéré. Voici les résultats très similaires des six méthodes de détection obtenus pour un exemple quelconque de signal stationnaire (seed = 5505361) :

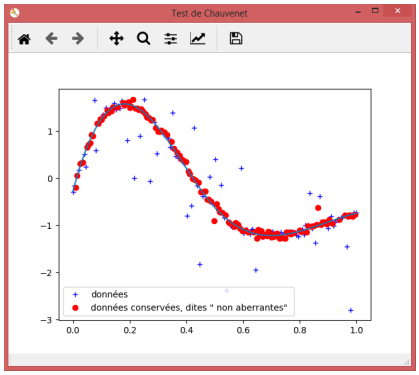


FIGURE 29 – Test de Chauvenet

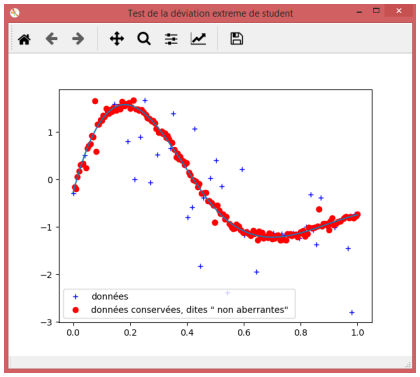


FIGURE 30 – Déviation extrême de Student

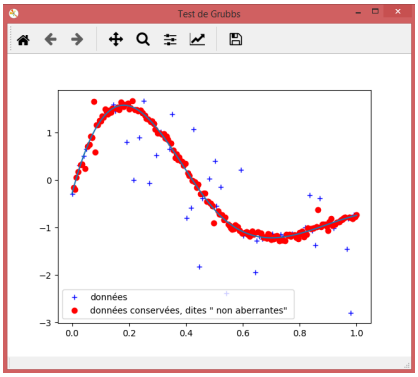


FIGURE 31 – Test de Grubbs

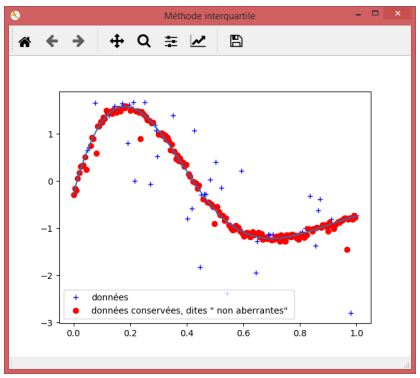


FIGURE 32 – Inter-quartile

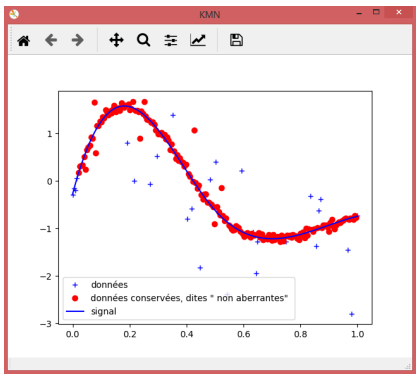


FIGURE 33 – k plus proches voisins

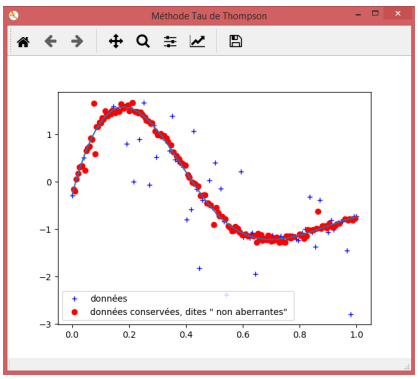


FIGURE 34 – Test tau de Thompson

3 RANSAC

4 LOESS

5 Comparaison

V
Produit rendu

1 Description de l'application

Pour améliorer l'accessibilité de nos méthodes, nous avons décider de toutes les regrouper en une seule application multi-fonction.

Elle se présente sous la forme de plusieurs fichier python, et l'utilisteur

VI
Conclusion

1 Bilan du travail réalisé

le projet initial a-t-il été suivi ? Comme souligné dans la sous-partie nommée Organisation en II-2, nous avons très rapidement dû modifier notre planification des tâches, nous avons rencontrés plusieurs difficultés durant ce stage applicatif : (ici on peut parler des difficultés rencontrées, les raisons de notre changement de planning / abandon des tâches / choix des solutions) PARLER DE LA TACHE 7 BILEL ET AMELYS

(puis un changement de paragraphe pour conclure sur les objectifs qui n'ont donc pas été atteints, à cause des difficultés rencontrées)

(puis un changement de paragraphe pour conclure sur les objectifs qui ont été atteints)

(la section bilan du travail d'équipe est pour l'instant un prototype qui je modifierai en fonction de ce qui sera mis dans cette partie)

2 Bilan du travail d'équipe

Concluons d'abord avec notre travail en équipe : Le confinement n'a pas été pratique, le télétravail à réduit cette sensation de convivialité que pourrait procurer un travail de groupe, et le partage d'écran pour faire comprendre son travail aux autres à montré ses limites. Mais nous avons tous oeuvré pour maintenir une bonne ambiance au sein du groupe. L'application de communication Discord nous a permis d'avoir une bonne circulation de l'information, puisque nous avions principalement des traces écrites de nos discussions et de nos décisions. Étant auteurs de notre planification des tâches et n'ayant pas ou peu d'expérience dans les projets de groupes sur plusieurs semaines, comme évoqué dans la partie II-2, nous sommes satisfait d'avoir garder en vue de livrer une application répondant à nos objectifs, au prix de certaines fonctionnalités. Cela est principalement dû à la fluidité de nos interactions sur Discord, qui ont permis de communiquer sur nos avancées, et donc de réaligner nos priorités pour satisfaire nos objectifs et d'en répartir les responsabilités.

3 Extensions envisagées

Nous avons écarté plusieurs idées durant le projet car nous manquions de temps. Nous tenons à les lister, afin de laisser des pistes de réflexion en cas de reprise du projet (par nous-même ou par d'autres personnes).

La première d'entre elles consiste en une autre méthode de création d'intervalles : une autre approche pourrait être de créer deux répartitions distinctes des données en intervalles. Cela signifierait que chaque donnée serait dans deux intervalles. Il faudrait ensuite appliquer les méthodes de détection des points aberrants sur les intervalles des deux répartitions, et ne considérer un point aberrant que s'il l'est pour les deux répartitions (sinon, cela signifierait que ce point est juste au bord d'un des deux intervalles.

Nous pourrions également chercher à automatiser la recherche des paramètres nécessaires, par exemple pour les méthodes de détection de points aberrants.

A ENLEVER APRES (images) On peut le voir sur les figures ci-dessous :

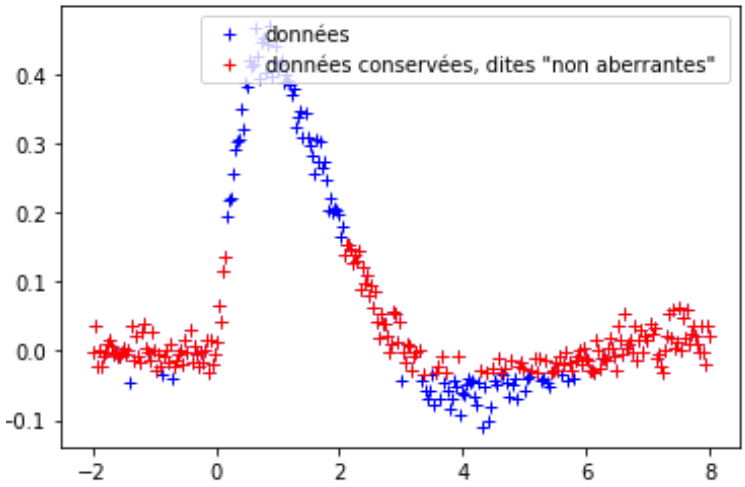


FIGURE 35 – Détection des points aberrants

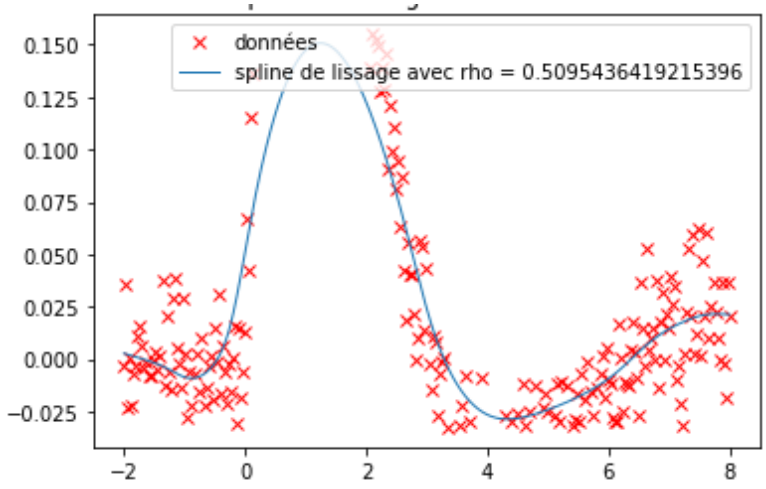


FIGURE 36 – Construction de la spline sans ces points

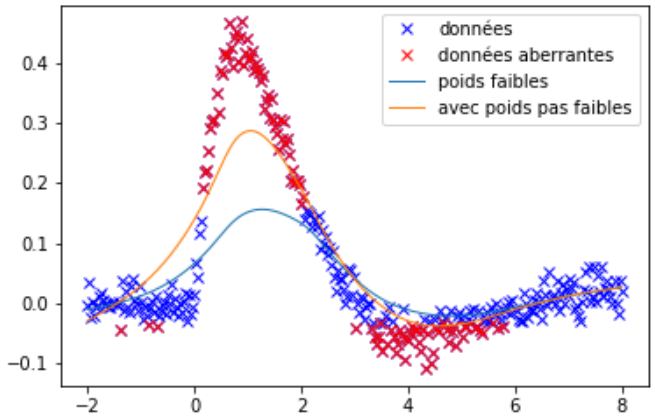


FIGURE 37 – Exemple de résultat

Dans cet exemple, nous avons en rouge une spline de lissage et en bleu une autre spline de lissage utilisant la méthode LOESS. Nous avons également les valeurs initiales de l'échantillon en rouge et celles ajustées en bleu. On remarque que la méthode LOESS considère moins les valeurs pouvant être considérées comme aberrantes ou bruitées.

[15] [1] [14] [4] [16] [17] [18] [10] [8] [9] [**MC**] [19] [20] [11] [12] [13]

VII
Bibliographie

Références

[1] Luc BIARD. *Polycopiés de cours distribués aux L3MI, en 2019/2020.* `http://www-ljk.imag.fr/membres/Luc.Biard/L3MI_cours/Splines.pdf`.

[2] *Estimation par noyau.* `https://fr.wikipedia.org/wiki/Estimation_par_noyau`.

[3] *Mean integrated squared error*.
https://en.wikipedia.org/wiki/Mean_integrated_squared_error.

[4] Rob J. Hyndman et George Athanasopoulos. *Forecasting : principles and practice*.
https://otexts.com/fpp2/ses.html, section 7.1 Simple exponential smoothing. 2014.

[5] Khadijetou El Heda. *Choix optimal du paramètre de lissage dans l'estimation non paramétrique de la fonction de densité pour des processus stationnaires à temps continu*.
https://tel.archives-ouvertes.fr/tel-02001263/document.

[6] Younès Benzaki. *Tout savoir sur les valeurs aberrantes*. https://mrmint.fr/outliers-machine-learning. 2017.

[7] *Identification des valeurs aberrantes avec la règle 1,5 x écart interquartile*. ://fr.khanacademy.org/mat
probability/summarizing-quantitative-data/box-whisker-plots/a/identifying-outliers-iqr-rule.

[8] *Grubbs' test for outliers*.
https://en.wikipedia.org/wiki/Grubbs%27s_test_for_outliers.

[9] *Test de Grubb*.
https://ellistat.com/guide-dutilisateur/statistiques-descriptives/tests-de-valeurs-aberrantes/test-de-grubb/.

[10] *Les tests pour la détection d'outliers*.
https://lemakistatheux.wordpress.com/2016/04/01/les-tests-pour-la-detection-doutliers/.

[11] Kenza Harifi. *Bien comprendre l'algorithme des k plus proches voisins (fonctionnement et implémentation sur R et Python)*.
https://medium.com/@kenzaharifi/bien-comprendre-lalgorithme-des-k-plus-proches-voisins-fonctionnement-et-impl%C3%A9mentation-sur-r-et-a66d2d372679. 2019.

[12] *K-nearest-neighbors algorithm*.
https://fr.qwe.wiki/wiki/K-nearest_neighbors_algorithm.

[13] Nicolas Turenne. *Méthode des KNN*.
http://exorciste2.free.fr/Amine/nouveau%20dossier/MOAB/coursDM_KNN.pdf. 2006.

[14] *RANSAC*.
http://w3.mi.parisdescartes.fr/~lomn/Cours/CV/SeqVideo/Material/RANSAC-tutorial.pdf. 2006.

[15] Luc Biard. *Documents de CAO*.
http://www-ljk.imag.fr/membres/Luc.Biard/#Teaching, section Algèbre linéaire pour le graphique et la CAO (L3 MI).

[16] Xavier Bourret Sicotte. *Locally Weighted Linear Regression (Loess)*.
https://xavierbourretsicotte.github.io/loess.html.

[17] *Local Regression*.
https://en.wikipedia.org/wiki/Local_regression.

[18] *Données aberrantes*.
https://fr.wikipedia.org/wiki/Donn%C3%A9e_aberrante.

[19] *Winsorizing*.
https://en.wikipedia.org/wiki/Winsorizing.

[20] Rick Wicklin. *Winsorization : the good, the bad and the ugly.* https://blogs.sas.com/content/iml/2017/02/08/winsorization-good-bad-and-ugly.html.

VIII
Annexes

1 Correspondance numéro/contenu des tâches

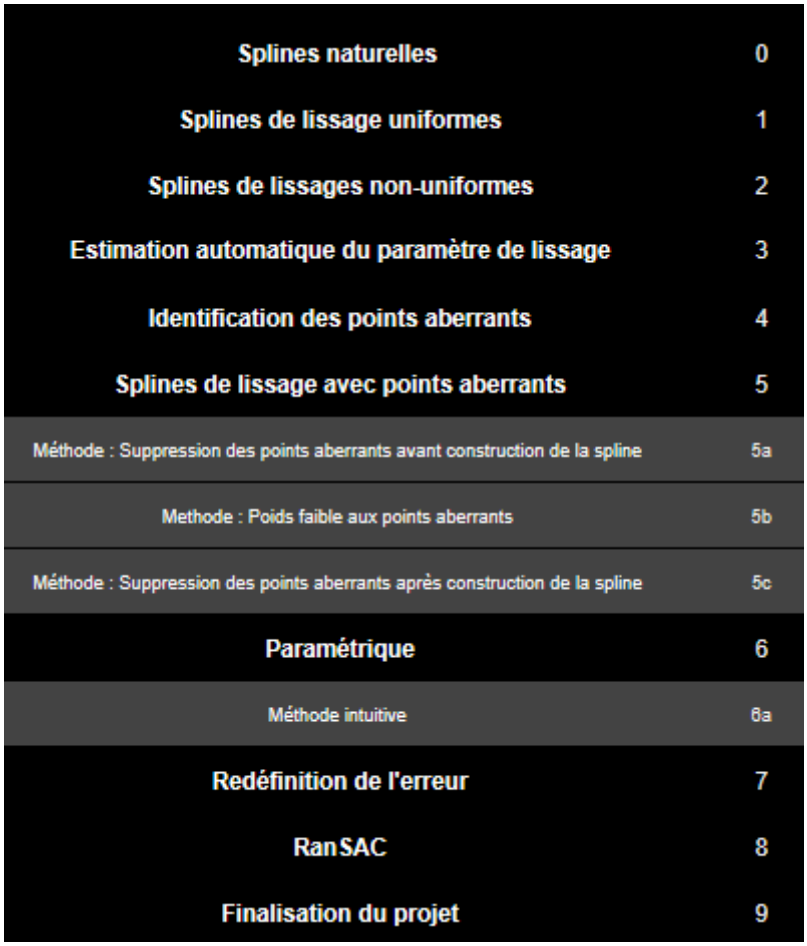


Figure 38 – Tâches prévues

Figure 39 – Tâches rajoutées

2 Répartition des tâches originelle

	lundi 6 avril		mardi 7 avril		mercredi 8 avril		jeudi 9 avril		vendredi 10 avril	
Amélys	0	0	5a	5a	5a	5a	5a		7	7
Béryl	4	4	4	4	4	6b	6b	6b	6b	6b
Bilel	2	2	2				8	8	8	8
Mohamed	3	3	3				3	3	-> 7	-> 7
Zakaria	-> 4	-> 4	-> 4	-> 4	-> 4			5c	5c	5c
Clément	1	1	5b	5b	5b	5b	5b	6a	6a	6a

FIGURE 40 – Répartition des tâches : semaine 1

mardi 14 avril		mercredi 15 avril		jeudi 16 avril		vendredi 17 avril	
7	7	7	7	7	7	7	7
6b	6b	6b	6b	6b	6b	6b	6b
8	8						
-> 7	-> 7	-> 7	-> 7	-> 7	-> 7	-> 7	-> 7
5c	6c	6c	6c	6c	6c	6c	6c

FIGURE 41 – Répartition des tâches : semaine 2

lundi 27 avril		mardi 28 avril		mercredi 29 avril		jeudi 30 avril
7	7	9c	9c	9c	9c	9c
6d	9b	9b	9b	-> 9c	-> 9c	-> 9c
-> 9a	-> 9a	-> 9b	-> 9b	-> 9b	-> 9c	-> 9c
-> 7	-> 7	-> 9c	-> 9c	-> 9c	-> 9c	-> 9c
-> 6d		-> 9c	-> 9c	-> 9c	-> 9c	-> 9c
9a	9a	-> 9c	-> 9c	-> 9c	-> 9c	-> 9c

FIGURE 42 – Répartition des tâches : semaine 3

	Responsable de la tâche assignée
	Aide pour la tâche assignée en priorité
	Disponible pour aider ou approfondir une tâche

FIGURE 43 – Répartition des tâches : légende

3 Répartition des tâches réelle