

NOTICE UTILISATEUR

MOLONARI 2023

Groupe terrain

laure.desorgues@etu.minesparis.psl.eu

lise.plisson-saune@etu.minesparis.psl.eu

francisco.garcia_weiss@etu.minesparis.psl.eu

tristan.montalbetti@etu.minesparis.psl.eu

zikang.zhu@etu.minesparis.psl.eu

marcoul.robin@etu.minesparis.psl.eu

Encadrement

agnes.riviere@minesparis.psl.eu

| | |
|---|-----------|
| OBJECTIF DU CODE..... | 2 |
| I - COMMENT UTILISER / MODIFIER LE CODE ?..... | 3 |
| 1. Les données brutes..... | 3 |
| 2. Les données de calibrations..... | 5 |
| 3. Rangement des données dans les dataframes..... | 6 |
| 4. Calcul de dH..... | 8 |
| II - COMPOSITION DU CODE..... | 10 |
| 1. Traitement des données brutes..... | 10 |
| 2. Décorrélation des données..... | 12 |
| 2.1. Filtre passe-bas..... | 12 |
| 2.2. Analyse d'ondelette..... | 14 |
| 2.3. Fast Fourier Transform (FFT)..... | 17 |
| 3. Inversion des données..... | 19 |
| 4. Tracé d'une carte avec les paramètres..... | 21 |
| 5. Analyse des données météorologiques..... | 23 |
| 6. Calibration des capteurs..... | 26 |
| 6.1. Calibration Voltage - Pression..... | 26 |
| 6.2. Calibration Voltage - Temperature..... | 27 |
| III - UTILISATION DU CODE ET INTERPRÉTATION DU RÉSULTAT..... | 29 |
| 1. Choix des Points Corrélos et Visualisation..... | 29 |
| 2. Résultat de la méthode FFT..... | 32 |
| 3. Interprétation des résultats..... | 33 |

OBJECTIF DU CODE

L'objectif est de récupérer les données brutes de pression et de températures et de les nettoyer afin qu'elles deviennent utilisables et ensuite effectuer une inversion pour récupérer les paramètres du terrain. Pour cela nous avons créé un code qui va exécuter les actions suivantes :

- Récupérer les données automatiquement depuis le dossier `raw_data`
- Les regrouper dans deux dataframes `pandas` que l'on nomme `pression` et `temperature`
- Enlever les colonnes inutiles et rajouter une colonne où l'on calcule `dH`
- Enlever les valeurs aberrantes en utilisant les méthodes IQR et Z Score
- Traiter les données pour décorreler `dH` et la température rivière en utilisant un filtre passe-bas, l'analyse d'ondelette et la méthode de Fast Fourier Transform (FFT)
- Effectuer l'inversion, récupérer les paramètres du terrain : la porosité, la conductivité, la perméabilité intrinsèque, la capacité thermique, leur distribution statistique, afficher les quartiles sur le modèle direct.

I - COMMENT UTILISER / MODIFIER LE CODE ?

1. Les données brutes

Il faut que les données soient rangées dans un fichier qui s'appelle `raw_data` qui doit se trouver à la même hauteur que le notebook. Si le fichier ne s'appelle pas comme ça il faut le modifier dans le tout début du code :

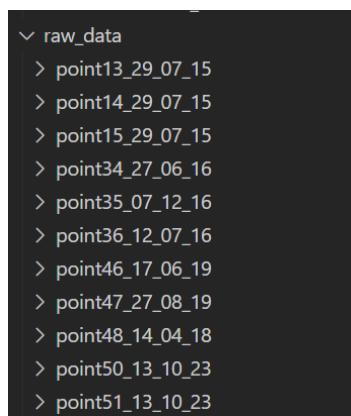
Récupérer les fichiers CSV:

```
# on récupère les informations du dossier
dossier = './raw_data'
contenuDossier = os.listdir(dossier)
print(contenuDossier)
nombrePoints = 0
```

Si le fichier n'est pas situé à cet endroit, il faut ajouter le chemin vers le fichier de la façon suivante :

Exemple : `'./MOLONARI/Projet/raw_data'`

A l'intérieur du dossier `raw_data`, il faut mettre un dossier par point que l'on veut traiter. Ces dossiers doivent s'appeler `pointXX_XX_XX_XX`, les deux premiers XX sont le numéro du points et ensuite il y a la date à laquelle le point a été implanté sous le format jour/mois/année. Par exemple :



Sur cette capture, le premier dossier `point13_29_07_15` correspond ainsi au point 13 qui a été mis dans la rivière le 29 juillet 2015. Il peut y avoir d'autres fichiers ou dossiers dans `raw_data`, tant que leur nom ne commence pas par 'point', cela ne générera pas le code.

Dans chaque dossier de points, il doit y avoir 3 fichiers :

- Le fichier .csv des températures
- Le fichier .csv des pressions
- Un fichier `geometrie.txt`

Pour être reconnu par le code, le nom du fichier csv des températures doit commencer par un `t` (/!\ un t minuscule) et finir par `.csv`. De même, le nom du fichier csv des pressions doit commencer par un `p` (/!\ un p minuscule) et finir par `.csv`.

Dans le fichier csv de la température il doit y avoir une colonne dates et autant de colonnes températures que de profondeurs énoncées dans `geometrie.txt` (dans la capture d'écran suivante il doit donc y en avoir 4).

Dans le fichier csv de la pression il doit y avoir une colonne date, une colonne pression et une colonne pour la température au niveau de la rivière.

Si il y a plus de colonnes que ça ce n'est pas grave, elles seront automatiquement supprimées des dataframes pandas par le code.

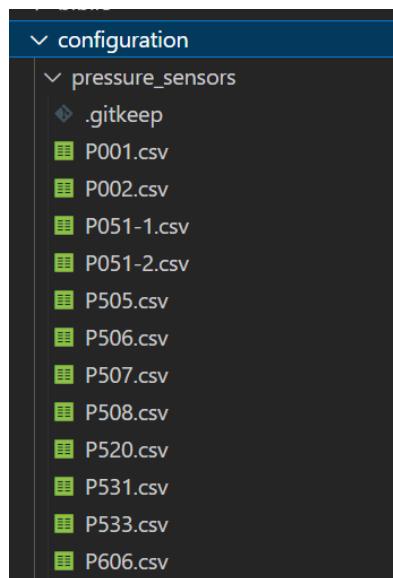
Le fichier `geometrie.txt` doit être écrit de la façon suivante :

```
raw_data > point13_29_07_15 >  ≡ geometrie.txt
 1  #pressure_sensor
 2  p505
 3  #depth river bed - river tube [cm]
 4
 5  #temperature_depths[cm]
 6  10;20;30;40
```

Ce qui est important c'est qu'il y ait le nom du capteur (attention ça commence par un p minuscule) en deuxième ligne et les profondeurs des températures séparées par un ; en ligne 6. Ce sont les deux informations que le code va aller chercher.

2. Les données de calibrations

Pour pouvoir calculer dH , il faut récupérer les valeurs `Intercept`, dU/dH et dU/dT . Ces valeurs doivent se situer dans un dossier `configuration` situé à la même hauteur que `raw_data` puis un deuxième dossier `pressure_sensors`. Dans ce sous-dossier, les données de calibrations doivent s'appeler `PXXX.csv` (/!\ un P majuscule cette fois) où XXX est le numéro du capteur.



Dans ces fichiers csv, il doit y avoir une ligne `Intercept`, un autre dU/dH et troisième dU/dT suivi d'une virgule et de la valeur correspondante. Il est important que ces trois mots soient écrits exactement de la même façon que ci-dessous.

```
configuration > pressure_sensors > P505.csv
1   Sensor_Name,P505
2   Datalogger,Raspberry PI
3   Calibration_Date,2016/08/12 10:10:10
4   Intercept,1.59324732652933
5   dU/dH,-8.89963363336198
6   dU/dT,-0.0136859629811579
7   Sigma_Meas_P,0.01
8
```

3. Rangement des données dans les dataframes

A l'intérieur du code, les données sont rangées dans une liste python `data`, cette liste contient des dictionnaires, un par point. Dans chaque dictionnaire, on peut accéder aux informations du point en utilisant les clés suivantes :

- `numero` : c'est le numéro du point
- `nom` : c'est le nom du dossier qui lui correspond
- `date` : c'est la date à laquelle il a été implanté
- `chemin` : c'est le chemin vers son dossier
- `capteur` : c'est le capteur qui a servi à faire la mesure (celui que l'on trouve dans `geometrie.txt`)
- `profondeur` : On ne se sert pas de cette valeur, elle correspond à laquelle se situe le capteur dans la rivière selon le fichier `geometrie.txt`. Dans tous les points que nous avons reçus elle n'était pas renseignée donc nous avons mis 0.
- `profondeurMesures` : C'est une liste des profondeurs auxquelles on a pris des mesures (!/ on ne prend pas en compte la température de la rivière qui se situe dans le fichier csv des pressions et qui correspondrait à une profondeur 0)
- `pression` : C'est un dataframes pandas qui a 3 colonnes : `dates`, `tension` et `temperature_stream`. Dans la suite du code on ajoute une colonne `dH`.

| | dates | tension | temperature_stream | dH |
|---|------------------|----------------|---------------------------|-----------|
| 0 | 12-07-2016 12:00 | 1.56471 | 20.007 | -0.222196 |
| 1 | 12-07-2016 12:15 | 0.62210 | 14.361 | 0.416653 |
| 2 | 12-07-2016 12:30 | 0.63614 | 14.314 | 0.407006 |
| 3 | 12-07-2016 12:45 | 0.69780 | 14.218 | 0.364748 |
| 4 | 12-07-2016 13:00 | 0.66911 | 14.098 | 0.384245 |

- **temperature** : C'est un dataframes pandas qui a une colonne **dates** et autant de colonnes températures que de profondeurs qui s'appellent **Temp_profondeur_XX** où XX est la profondeur en cm.

| | dates | Temp_profondeur_10 | Temp_profondeur_20 | Temp_profondeur_30 | Temp_profondeur_40 |
|-----|------------------|---------------------------|---------------------------|---------------------------|---------------------------|
| 198 | 15-10-2023 11:15 | 12.251 | 13.102 | 13.617 | 14.018 |
| 199 | 15-10-2023 11:30 | 12.251 | 13.094 | 13.601 | 14.005 |
| 200 | 15-10-2023 11:45 | 12.263 | 13.087 | 13.586 | 13.990 |
| 201 | 15-10-2023 12:00 | 12.252 | 13.079 | 13.571 | 13.976 |
| 202 | 15-10-2023 12:15 | 12.245 | 13.072 | 13.557 | 13.963 |

A la fin du code, il y aura deux nouvelles clés dans les dictionnaires :

- **pression2** : C'est le même dataframe que **pression** mais on a enlevé les valeurs aberrantes.
- **température2** : Même chose pour les températures.

Attention : Les dataframes **pression** et **temperature** sont étroitement liés, il faut que leurs colonnes **dates** soient exactement les mêmes. Autrement dit, si on supprime une valeur pour pression, il faut également supprimer la valeur de température qui correspond à la même date et inversement.

Un exemple de dictionnaire ci-dessous.

```

data[5]
[9]
...
{'numero': '13',
 'nom': 'point13_29_07_15',
 'date': '29_07_15',
 'chemin': './raw_data/point13_29_07_15',
 'capteur': 'p505',
 'profondeur': 0,
 'profondeurMesures': [10, 20, 30, 40],
 'pression':      # Date Heure, GMT+02:00 \
 0      1 07/23/15 02:00:00 PM
 1      2 07/23/15 02:15:00 PM
 2      3 07/23/15 02:30:00 PM
 3      4 07/23/15 02:45:00 PM
 4      5 07/23/15 03:00:00 PM
 ...
 ...
 558 559 07/29/15 09:30:00 AM
 559 560 07/29/15 09:45:00 AM

```

Pour le fonctionnement du code, nous avons décidé de le commenter le plus possible lorsque cela était nécessaire. Pour les étapes plus délicates, nous avons écrit une explication ci-dessous.

4. Calcul de dH

D'après la calibration du signal avec prise en compte de l'effet de la température, on obtient la relation ci-dessous:

$$dH = \frac{dH}{dU} (V - Intercept - \frac{dU}{dT} T_{riv})$$

où `k0 = Intercept`, `k1 = dU/dH`, `k2 = dU/dT` sont trois paramètres qui dépendent de l'étalonnage du capteur. En connaissant les données de la tension, de la température rivière et ces trois paramètres du capteur, on peut alors calculer la charge dH par:

$$dH = \frac{1}{k_1} (V - k_0 - k_2 T_{riv})$$

II - COMPOSITION DU CODE

1. Traitement des données brutes

Le code en Python traite les données brutes provenant de capteurs de pression et de température de manière à ce qu'elles soient utilisables. Voici une explication étape par étape :

Importation de bibliothèques : telles que `pandas`, `numpy`, `matplotlib`, `seaborn`, `datetime`, `chardet`, `csv`, `os`, `matplotlib.dates`, `Image`, et `scipy.stats`.

Récupération des informations du dossier 'raw_data' : le script liste le contenu du dossier 'raw_data' et compte le nombre de fichiers qui commencent par 'point'. Il récupère également les noms et numéros de ces fichiers.

Fonction `read_csv` définie pour lire un fichier CSV en détectant automatiquement le séparateur et en sautant la première ligne si elle contient le mot "Titre".

Construction d'une structure de données :

- Le code crée un dictionnaire `data` qui stocke les informations sur chaque point (capteur) trouvé dans le dossier.
- Pour chaque point, il récupère des informations telles que le numéro, le nom, la date, le chemin vers le dossier, le capteur, la profondeur du capteur, les profondeurs auxquelles les mesures sont effectuées, et charge les données de pression et de température depuis les fichiers correspondants.

Nettoyage des données :

- Les colonnes inutiles sont supprimées des *DataFrames* de pression et de température.
- Les colonnes sont renommées de manière appropriée.
- Les dates sont converties dans un format approprié.

```

# Conversion dans le bon format date
for x in data.values():
    x['pression']['dates'] = x['pression']['dates'].apply(lambda x: parser.parse(x).strftime('%d-%m-%Y %H:%M'))
    x['temperature']['dates'] = x['temperature']['dates'].apply(lambda x: parser.parse(x).strftime('%d-%m-%Y %H:%M'))
#    point['pression']['dates'] = pd.to_datetime(point['pression']['dates'], format='mixed')
#    point['temperature']['dates'] = pd.to_datetime(point['temperature']['dates'], format='mixed')

```

Merge des données de pression et de température : les DataFrames de pression et de température sont fusionnés sur la colonne des dates.

Traitement des données aberrantes (méthode Z-score) :

- Pour chaque point, le script calcule le Z-score pour chaque colonne de données.
- Les données dont les Z-scores dépassent un certain seuil sont exclues.

```

for x in data.values() :
    # Traiter chaque colonne sauf la colonne des dates
    columns_to_process = [col for col in x['to_process'].columns if col != 'dates']
    # Créer un DataFrame vide pour stocker les données traitées
    x['processed'] = x['to_process'][['dates']].copy()
    # Boucler à travers chaque colonne à traiter
    for column_name in columns_to_process :
        # Calculer le Z-score pour la colonne
        z_scores = np.abs(stats.zscore(x['to_process'][column_name]))
        # Définir un seuil pour le Z-score (par exemple, 3)
        threshold = 3
        # Sélectionner les lignes avec des Z-scores inférieurs au seuil
        df_cleaned = x['to_process'][z_scores < threshold]
        # Copier les données traitées dans df_processed
        x['processed'][column_name] = df_cleaned[column_name]

```

Explication de la méthode Z-Score :

- **Définition d'un seuil :** Un seuil est choisi (ici c'est 3). Cela signifie que toutes les données ayant un Z-score supérieur à 3 ou inférieur à -3 seront considérées comme des valeurs aberrantes.
- **Calcul des Z-scores pour chaque point de données :** Pour chaque colonne de données, le script calcule le Z-score pour chaque point de données.
- **Identification des valeurs aberrantes :** Les points de données dont les Z-scores dépassent le seuil défini sont considérés comme des valeurs aberrantes.

- **Exclusion des valeurs aberrantes** : Les données correspondantes aux valeurs aberrantes identifiées sont exclues du jeu de données traité.

Nettoyage final des données : les lignes contenant des valeurs nulles sont supprimées.

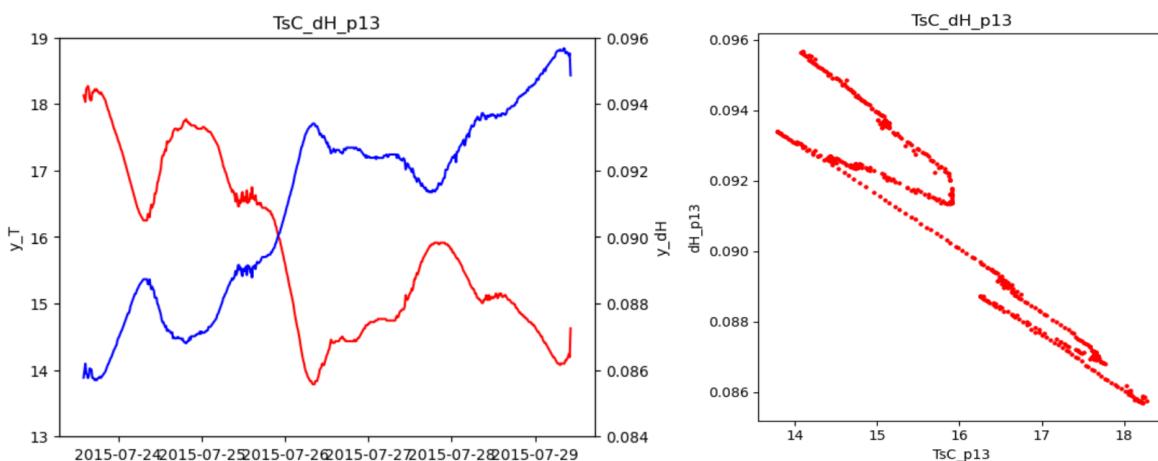
```
for x in data.values() :
    x['processed'] = x['processed'].dropna(axis = 0, how = 'any')
```

2. Décorrélation des données

La calibration du signal montre que la charge dH est affectée par la température, mais certaines variations de température ont un effet indésirable sur la charge (influence excessive). Par exemple, la variation de température dû à l'alternance des saisons doit être prise en compte, mais la différence de température entre le matin et le soir d'une journée doit être ignorée. Par conséquent, le but de cette partie est de supprimer l'effet des variations à court terme de température sur la charge.

2.1. Filtre passe-bas

Ici nous prenons l'exemple du [Point13](#), qui sera montré par la suite d'être le point le plus corrélé. On prend en entrée les données de la charge dH et de la température rivière [TsC](#) pour faire le dessin et le nuage de corrélation. D'après ces deux figures, c'est évident que [dH](#) et [TsC](#) sont corrélées presque tout le temps.



Pour décorreler les données de ce point, tout d'abord, nous appliquons un Butterworth filtre passe-bas pour filtrer les vibrations à haute fréquence dans `dH` entraînées par les variations à court terme de température.

```
# Butterworth filter method

from scipy import signal

data_to_filter = dH
order = 2
cutoff_freq = 2*(15/60/30)

b, a = signal.butter(order, cutoff_freq, 'lowpass')

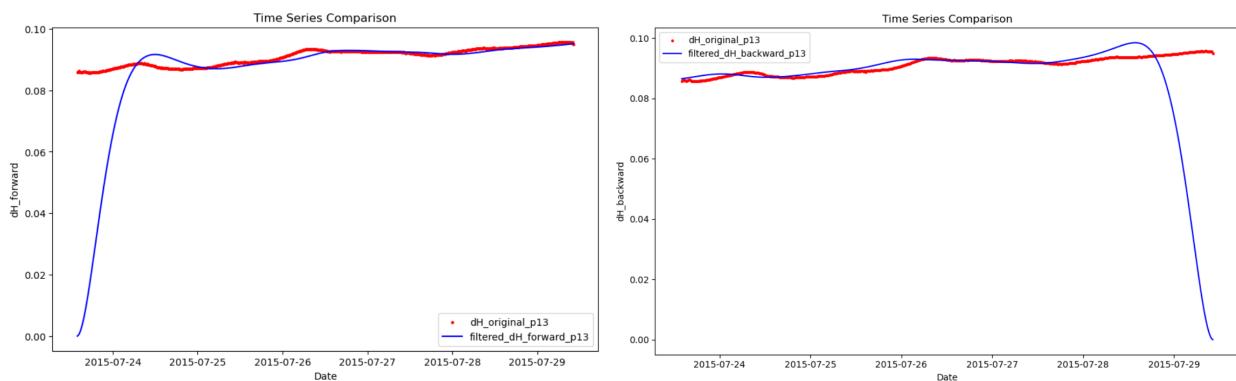
filtered_data_low = signal.lfilter(b, a, data_to_filter)
```

Il marche bien mais le problème est que le filtre fonctionne avec une période d'initialisation à partir de valeur 0. Durant cette période, les données ne sont pas vraiment post-filtrées. Pour le résoudre, nous pouvons appliquer ce filtre deux fois, dans le sens direct et inverse.

```
# Butterworth two-way-filter method

forward_filtered_signal_low = signal.lfilter(b, a, data_to_filter)

reversed_signal = data_to_filter[::-1]
backward_filtered_signal = signal.lfilter(b, a, reversed_signal)
backward_filtered_signal_low = backward_filtered_signal[::-1]
```



Ensuite, nous pouvons combiner les résultats à l'aide d'une fonction de poids, de sorte que toutes les données obtenues peuvent être considérées comme post-filtrées. Par exemple, dans le résultat filtré directement, le début est aberrant, donc on doit prendre en compte plutôt de cette partie du résultat filtré inversement, et vice versa.

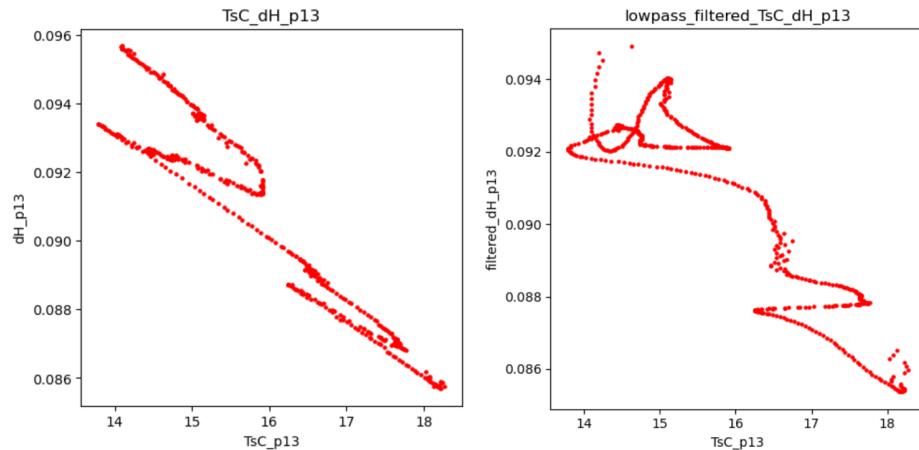
```
# Combination of two-way-filtered signal with a weight fonction

L = len(data_to_filter)
weight = np.zeros(L)
combined_filtered_signal_low = np.zeros(L)

a = 1.0 / L
for i in range(L):
    weight[i] = i * a

for i in range(L):
    combined_filtered_signal_low[i] = forward_filtered_signal_low[i] * weight[i] + backward_filtered_signal_low[i] * (1-weight[i])
```

En comparant le nuage de corrélation des signaux original et filtré, on peut en déduire que le filtre passe-bas est utile pour décorrélérer **dH** et **TsC**, mais il existe encore le phénomène de corrélation.



2.2. Analyse d'ondelette

Afin de poursuivre le but de décorrélation, nous essayons d'appliquer la méthode de la transformation en ondelettes pour analyser les caractéristiques fréquentielles temporelles de **dH** et **TsC**. Nous choisissons ici la fonction morlet pour l'opération de transformation en ondelettes continues (Continuous Wavelet Transform, CWT).

```

# Ondelette analysis of TsC and dH

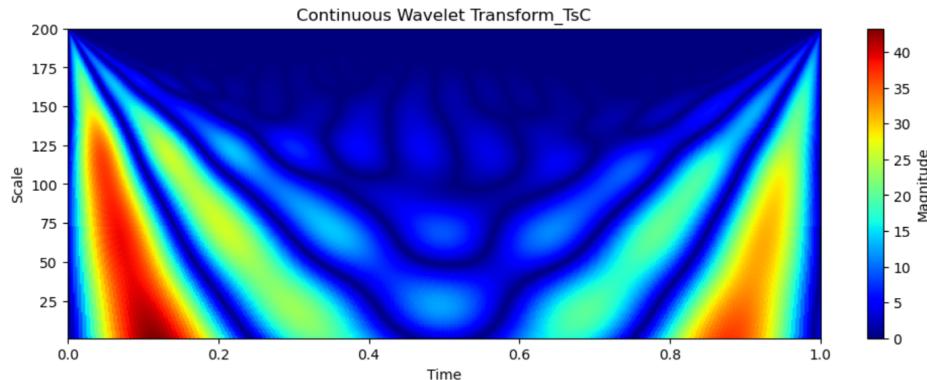
import pywt

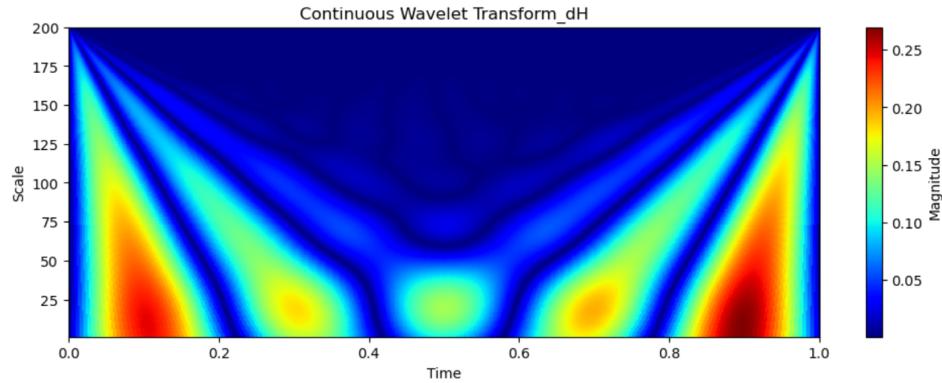
wavelet = 'morl' # Choose "Morlet" as basis function
scales = np.arange(1, 200)

# Continuous Wavelet Transform (CWT)
coeffs_TsC, freqs = pywt.cwt(TsC, scales, wavelet)
coeffs_dH, freqs = pywt.cwt(dH, scales, wavelet)

```

Ensuite, on trace et analyse la figure des coefficients de la transformation en ondelettes continues. On constate que les deux figures sont très similaires, ce qui montre une fois de plus que les données originales de **dH** et **TsC** du **Point13** sont très corrélées. Dans les figures, le signal dominant a une plus grande magnitude, ce qui correspond à la région de couleur chaude. On peut donc en déduire que le signal dominant de **TsC** apparaît 5 fois dans la figure de **dH**. En considérant que la durée totale de la collecte de données est d'environ 5 jours, cela signifie que la période du signal d'influence dominante de **TsC** sur **dH** est d'environ 1 jour.





Avec ce résultat, de manière plus ciblée de décorrélation, nous pouvons appliquer un filtre coupe-bande. Nous filtrons les signaux avec une période d'environ 1 jour dans `dH` pour éliminer l'influence dominante de `TsC`.

```
# Using a bandstop filter to remove the frequency of the dominant influence of TsC on dH

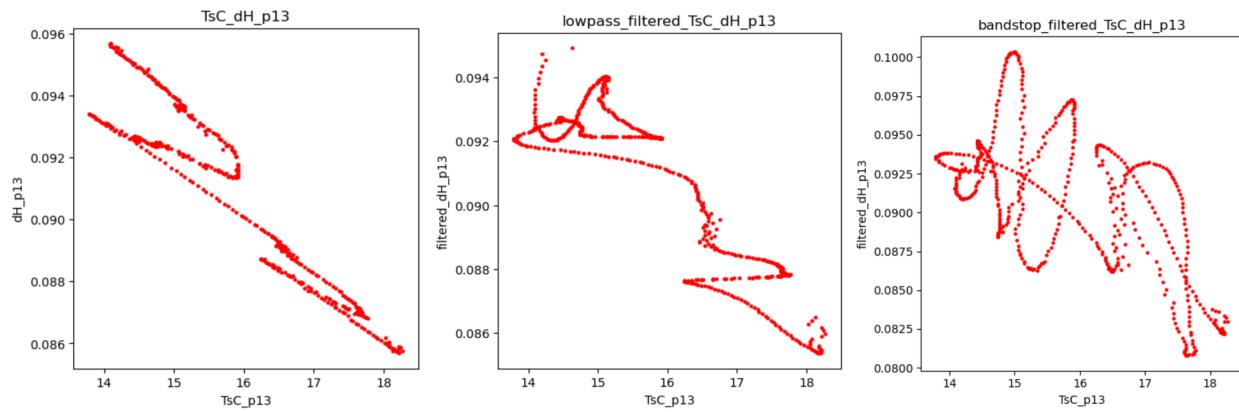
from scipy.signal import butter, lfilter

fs = 1/(15*60) # one data every 15min

def butter_bandstop_filter(data, lowcut, highcut, fs, order=4):
    nyquist = 0.5 * fs
    low = lowcut / nyquist
    high = highcut / nyquist
    b, a = butter(order, [low, high], btype='bandstop')
    y = lfilter(b, a, data)
    return y

lowcut = 1/26/3600
highcut = 1/22/3600
```

Comme précédemment, avec ce filtre coupe-band, nous appliquons le filtrage bi-directionnel et la fonction de poids pour obtenir le signal de `dH` complètement post-filtré. Finalement, en traçant le nuage de décorrélation et le comparons avec les deux nuages précédents, nous constatons que le signal `dH` obtenu par le filtre coupe-bande est le plus décorrélé. Nous pouvons donc conclure que la méthode de la transformation en ondelettes fonctionne mieux.



2.3. Fast Fourier Transform (FFT)

Transformée de Fourier Rapide (FFT), une technique algorithmique utilisée pour calculer efficacement la transformée de Fourier discrète d'un signal. La FFT permet d'analyser les composantes fréquentielles d'un signal en décomposant celui-ci en une somme de sinus et de cosinus, représentant différentes fréquences.

La méthode FFT est celle retenue dans le traitement des différents signaux de dH. Le code de cette méthode se situe dans le notebook : [traitement_data_fft_molonari.py](#) à la fin du notebook.

Dans le contexte de ce code, la FFT est utilisée pour analyser le signal de pression ([data_sample_dH](#)). Après avoir calculé la FFT avec `fft(data_sample_dH)`, le code extrait les fréquences associées et filtre spécifiquement celles qui correspondent à une période d'environ une journée. Les fréquences sélectionnées sont mises à zéro dans la FFT, ce qui revient à éliminer ces composantes fréquentielles du signal d'origine. Enfin, la transformée de Fourier inverse ([ifft](#)) est utilisée pour reconstruire le signal filtré.

La FFT est couramment utilisée dans le traitement du signal pour des applications telles que la suppression du bruit, l'analyse spectrale et la détection de motifs périodiques. C'est le cas dans notre projet où l'on cherche à éliminer l'influence de la périodicité journalière de la température sur la différence de charge dH. La méthode FFT offre une approche efficace pour représenter un signal dans le domaine fréquentiel, permettant ainsi de mieux manipuler et d'éliminer les fréquences souhaitées: ici la fréquence de $f = \frac{1}{24 \times 3600} \text{Hz}$.

```

# Calculer la transformée de Fourier du signal audio
fft_result_dH = fft(data_sample_dH)
frequencies_dH = fftfreq(len(fft_result_dH), d=1/sample_rate)
# Sélection des fréquences entre ayant une période d'environ 1 journée
mask = np.where((frequencies_dH > 0.0385) & (frequencies_dH < 0.045), True, False)
# On récupère les indices des fréquences sélectionnées
interval_peaks_half = np.where(mask)[0]
# Ajout des fréquences symétriques dans la liste des fréquences à éliminer
interval_peaks=list(interval_peaks_half)
for l in interval_peaks_half:
    interval_peaks.append(len(mask)-l)
# Copie du signal fft_result_dH
fft_result_dH_copy=fft_result_dH.copy()
# On met à 0 les fréquences sélectionnées (les fréquences à éliminer)
for l in interval_peaks:
    fft_result_dH_copy[l]=0
# On reconstruit le signal en appliquant la transformée de Fourier inverse
signal_reconstruit_dH = ifft(fft_result_dH_copy)

```

Boucle sur les Points : Le code utilise une boucle `for` pour itérer sur les différents points, représentés par `j` dans la variable `numeroPoints`.

Vérification de la Corrélation : Pour chaque point, le code vérifie s'il est corrélé en utilisant la condition `if j in point_correler`. Si le point est corrélé, le code continue ; sinon, il passe au point suivant.

Traitements des Données de Pression : Les données de pression du point actuel (`data_sample_dH`) sont extraites, ainsi que les dates correspondantes (`data_abscisses`).

Calcul de la Fréquence d'Échantillonnage : La fréquence d'échantillonnage est calculée en fonction d'un intervalle de temps fixe (15 minutes dans ce cas).

Traitements de Cas Particuliers : Certains points ont des cas particuliers où seule une partie des données doit être traitée. Cela est géré avec des conditions spécifiques (`if j=='35': et elif j=='47' or j=='46':`).

Transformée de Fourier (FFT) : Le code utilise la bibliothèque `scipy.fft` pour calculer la transformée de Fourier du signal de pression (`fft(data_sample_dH)`). Cela permet d'analyser les fréquences présentes dans le signal.

Filtrage des Fréquences : Le code sélectionne un intervalle de fréquences spécifique, correspondant à une période d'environ une journée, et met à zéro les fréquences situées dans cet intervalle.

Transformée Inverse de Fourier (iFFT) : Le signal est reconstruit en utilisant la transformée inverse de Fourier (`ifft(fft_result_dH_copy)`), éliminant ainsi les fréquences indésirables.

Visualisation des Résultats : Des graphiques sont générés pour visualiser le signal original, le signal traité, et l'évolution des graphes de corrélations avant et après traitement en fonction du point et des cas particuliers.

Exportation des Données Traitées : Les données de pression traitées sont exportées vers un fichier CSV portant le nom du point (`pointX_pression_traité.csv`). De même, les données de température traitées sont exportées vers un fichier CSV portant le nom du point (`pointX_temperature_traité.csv`).

Déplacement des Fichiers : Les fichiers CSV sont déplacés vers un dossier spécifique (`data_traité`).

Affichage et Contrôle : Des messages sont affichés pour indiquer si un point est corrélé ou non. La longueur des données de pression et de température est vérifiée à des fins de contrôle, et des graphiques supplémentaires sont générés pour certains cas particuliers.

3. Inversion des données

Après avoir nettoyé les données et appliqué le filtre pour retirer la corrélation entre température et pression, le code exporte les données propres en format csv sous forme

d'un fichier contenant les données de pression et de température de la rivière et un autre contenant ceux de température de l'aquifère. Il faut ensuite fournir ces données au code effectuant l'inversion situé dans `inversion/inversion.py`

On rentre les données du code dans `capteur_rivière` (pour la pression et `T_riv`) et dans `capteur_ZH` pour les températures.

```
19 capteur_riviere = pd.read_csv("data_traite/point51_pression_traité.csv", sep = ',', encoding='latin1')
20 capteur_ZH = pd.read_csv("data_traite/point51_temperature_traité.csv", sep = ',', encoding='latin1')
```

Le code effectue une première fois le calcul du modèle direct avec des paramètres standard. On voit s'afficher différents graphes contenant des frises temporelles de températures et de flux. Ensuite le code commence l'inversion par la méthode dream MCMC. Cette inversion peut être plus moins longue selon le nombre de simulations, de chaînes, la variance de perturbation et la plage de paramètres dans la distribution uniforme à priori. On peut modifier toutes ces données à partir de la ligne 167 du code :

```
167 priors_couche_1 = {
168     "moinslog10K": ((6, 11), .01), # (intervalle, sigma)
169     "n": ((.001, .5), .005),
170     "lambda_s": ((.1, 9), .1),
171     "rhos_cs": ((5e6,5e7), 1e5),
172 }
173
174
175 all_priors = [
176     ['Couche 1', 0.4, priors_couche_1]
177 ]
178
179 col.compute_mcmc(
180     nb_iter = 500,
181     all_priors = all_priors,
182     nb_cells = 100,
183     sigma2=1.0,
184     nb_chain=10
185 )
```

On remarquera que le code propose de prendre en compte plusieurs couches mais cela ne fonctionne pas sur cette version du code. En effet le groupe calculs n'ayant proposé une version fonctionnelle du modèle multicouche qu'un jour avant le rendu nous n'avons pas pu intégrer cette fonctionnalité dans nos simulations. Il faudrait intégrer leur version du code pour utiliser le modèle multi couche.

Dans la capture précédente on peut modifier l'intervalle de paramètres ainsi que le sigma dans `prior_couche_1`. On peut aussi modifier le nombre d'itération et de chaîne dans `compute_mcmc`

Le code affiche ainsi dans l'ordre différents graphiques que nous pourrons enregistrer :

- La distribution des paramètres
- Les mesures des capteurs et le modèle directe calculé avec les meilleurs paramètres
- Un grand graphe avec plusieurs frises temporelles de températures, de flux d'énergie et d'eau
- Des frises de flux calculés à partir des quantiles de distribution
- Les mesures et le modèle direct calculé avec les quantiles de distribution
- Le débit d'eau du modèle et des quantiles à la surface

Nous avons enregistré ces graphes dans le dossier `inversé` pour chaque point que nous avions.

Le code affiche aussi la valeur des meilleurs paramètres ainsi que la RMSE globale et pour chaque capteur qui permet de regarder l'erreur du modèle par rapport aux mesures. Nous avons également sauvegardé ces informations dans les fichiers `infos.md` pour chaque point.

Pour des besoins pratiques nous avons enregistré les meilleurs paramètres pour chaque point dans le fichier `inversion/params.csv` ainsi que le régime (infiltrant ou exfiltrant) majoritaire pour chaque point. Ce fichier a été rempli manuellement vu le faible nombre de points mais on pourrait imaginer un code capable de le générer automatiquement.

4. Tracé d'une carte avec les paramètres

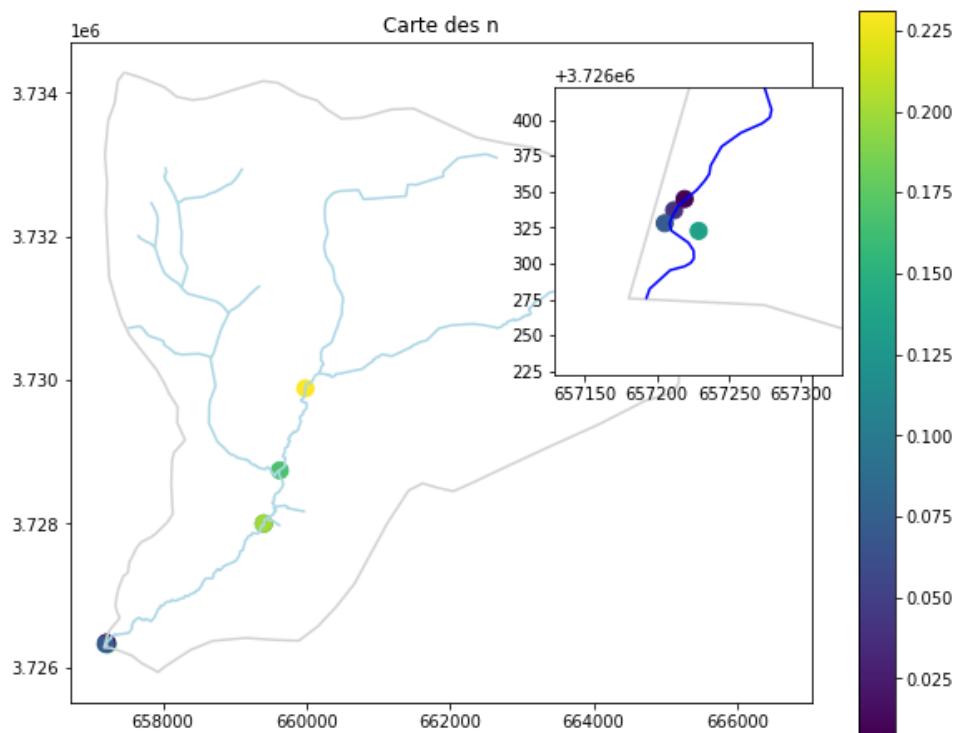
Dans le fichier map se trouve de quoi tracer une carte des paramètres inversés ainsi que les cartes tracés cette année. Le notebook `map.ipynb` trace des cartes de l'évolution de chaque paramètres.

Le code prend en entrée un dictionnaire avec la localisation géographique et la date de récolte de chaque point.

```
# Coordonnées géographiques d'origine
points_data = [
    {'name': 'point13_29_07_15', 'id': 13, 'latitude': 48.82745, 'longitude': 3.11332, 'date':'29/07/2015'},
    {'name': 'point14_29_07_15', 'id': 14, 'latitude': 48.82758, 'longitude': 3.11309, 'date':'29/07/2015'},
    {'name': 'point34_28_06_16', 'id': 34, 'latitude': 48.842323, 'longitude': 3.143102, 'date':'28/06/2016'},
    {'name': 'point35_03_08_16', 'id': 35, 'latitude': 48.85914, 'longitude': 3.15124, 'date':'03/08/2016'},
    {'name': 'point48_14_04_18','id': 48, 'latitude': 48.8489348, 'longitude': 3.1462324, 'date':'14/04/2018'},
    {'name': 'Point46_17_06_19','id': 46, 'latitude': 48.842323, 'longitude': 3.143102, 'date':'17/06/2019'},
    {'name': 'Point50_13_10_23','id': 50, 'latitude': 48.82765, 'longitude': 3.11319, 'date':'13/10/2023'},
    {'name': 'Point51_13_10_23','id': 51, 'latitude': 48.82750, 'longitude': 3.11300, 'date':'13/10/2023'}
]
```

Il faut s'assurer que les id de chaque point correspondent à ceux dans `params.csv`. Ce fichier est également pris en entrée du notebook.

Les cartes possèdent un petit zoom sur le ru les Avenelles où se trouvent plusieurs points, zoom sur le coin en bas à droite de la carte principale. Le notebook trace aussi une dernière carte qui associe chaque point à sa date.



Exemple de carte de la variation de la porosité du milieu dans le bassin de l'orgeval.

5. Analyse des données météorologiques

Le code permet de traiter les données météorologiques (précipitations et températures) à partir de fichiers texte et d'obtenir des graphes pour les analyser. Le code utilise les bibliothèques pandas, matplotlib, seaborn pour la manipulation des données et la visualisation graphique.

Les différentes étapes sont :

Chargement des données :

- Les chemins des fichiers texte contenant les données de pluie et de température sont définis.
- Les fichiers CSV sont chargés dans des tables de données (DataFrames) à l'aide de la bibliothèque pandas.

Nettoyage des données :

- Les colonnes 'Qualite', 'Min' et 'Max' sont supprimées des DataFrames pluie et temp.
- Les valeurs aberrantes (-9999) sont supprimées des colonnes 'Valeur' dans les deux DataFrames.

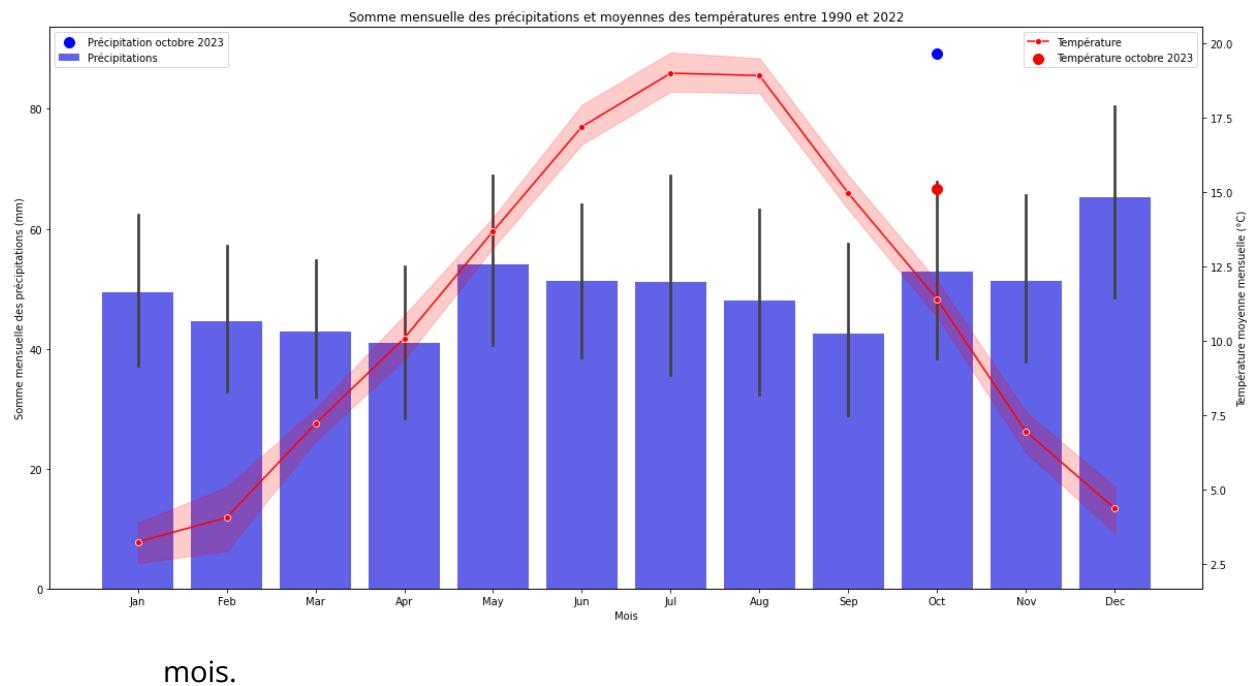
Conversion des dates : les colonnes de dates 'DateHeure' dans les deux DataFrames sont converties au format datetime.

Affichage des données et graphiques :

- Les DataFrames pluie et temp sont affichés.
- Deux graphiques sont générés : un pour afficher la quantité de pluie au fil du temps et un autre pour la température.

Analyse mensuelle :

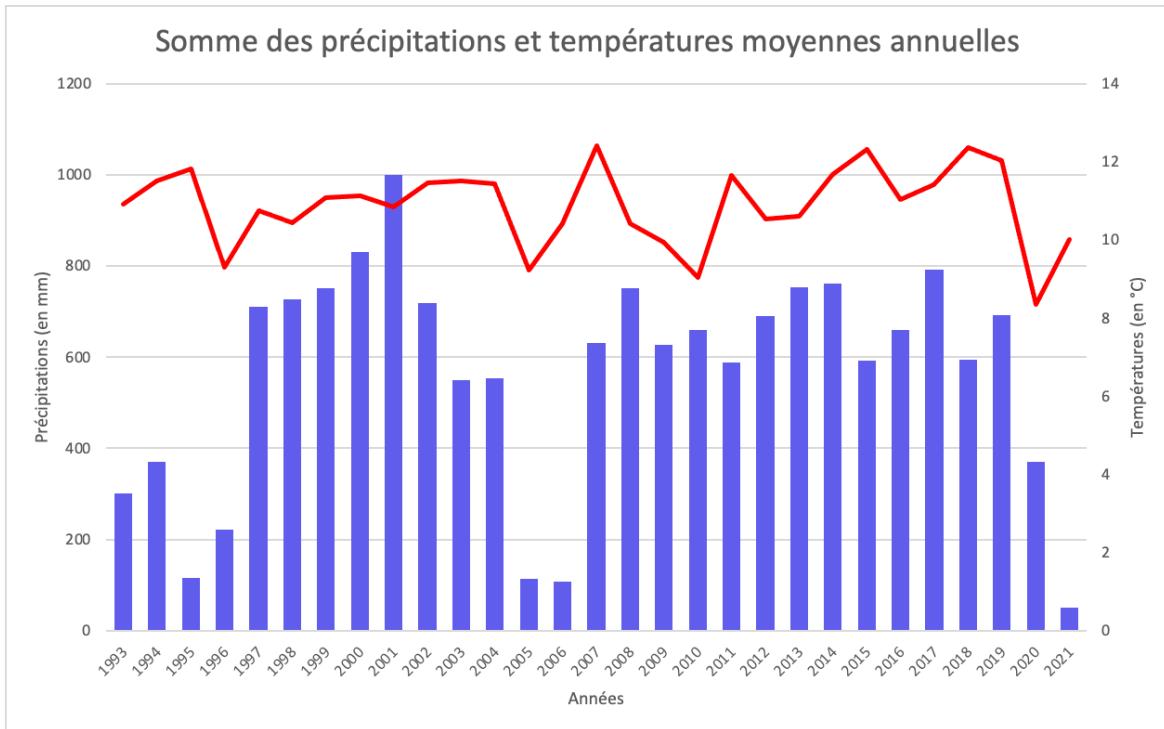
- Une colonne 'Mois' est créée dans le DataFrame temp, et une colonne 'Annee' est également ajoutée.
- Les données sont filtrées pour inclure uniquement les années civiles complètes.
- La moyenne des températures par mois pour toutes les années est calculée, et un DataFrame moyenne_mensuelle_temperature est créé.
- Un graphique en barres et en ligne est généré pour afficher la somme mensuelle des précipitations et la moyenne des températures pour chaque



Analyse annuelle :

- Une colonne 'Annee' est créée dans chaque DataFrame.
- La somme des précipitations par année et la moyenne des températures par année sont calculées.
- Les deux tableaux sont fusionnés sur la colonne 'Annee'.

- Un tableau combiné est affiché et un graphique combiné est généré pour montrer la somme annuelle des précipitations et la moyenne annuelle des températures depuis 1990.



6. Calibration des capteurs

On a suivi le protocole dans le lien suivant:

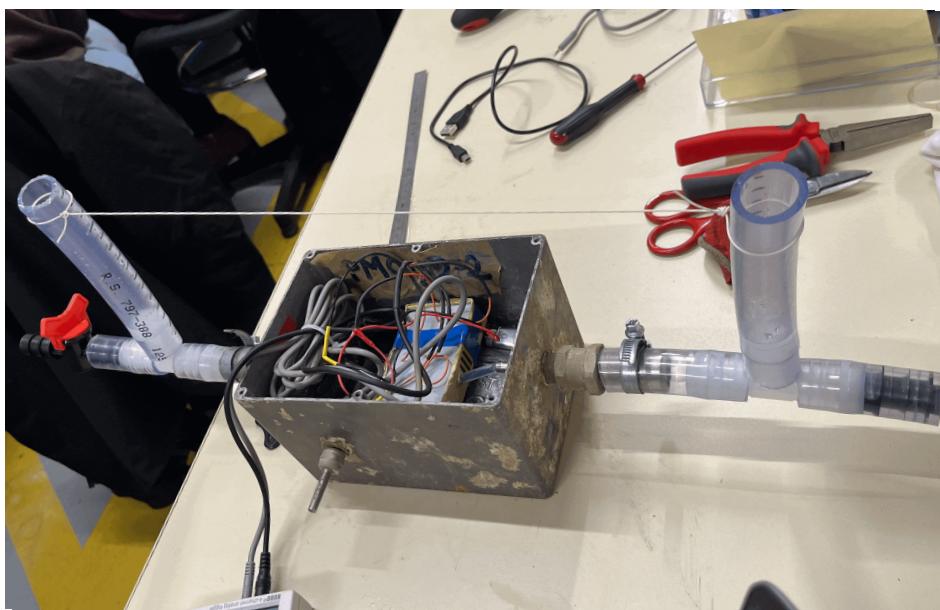
https://github.com/agnes-riviere/calibration_molonari_mini/blob/main/protocole_calibration.md

Ce processus permet d'obtenir les constants 'dU/dH', 'dU/dT' et 'Intercept' qui servent à transformer la tension mesurée sur le terrain en différence de charge.

L'objectif est de simuler des conditions que l'on peut trouver dans le milieu naturel: cas infiltrant et exfiltrant, variation journalière de température. Il faut s'assurer de travailler dans le rang de températures (5 - 25 °C) et différence de charge qui existent dans le cours d'eau et dans la nappe.

6.1. Calibration Voltage - Pression

- Préparation du dispositif comme dans la photo

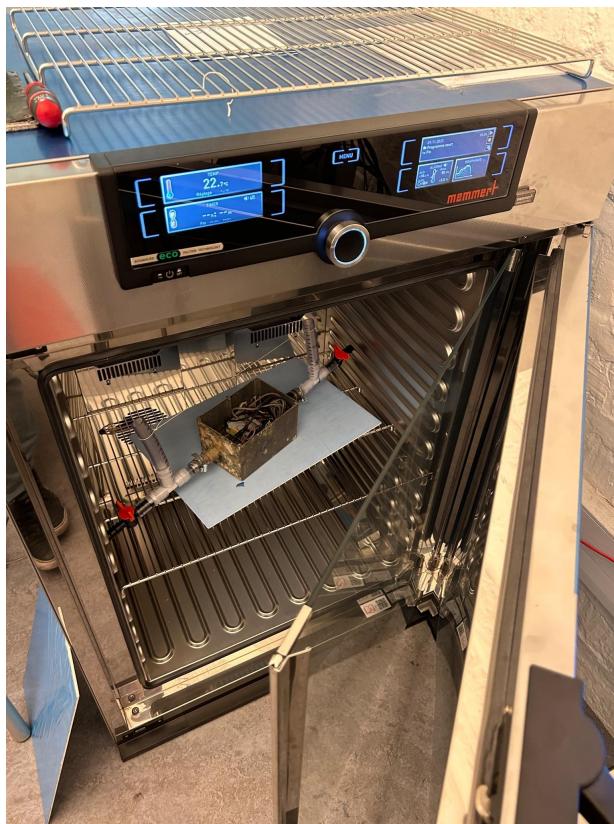


- Mesure de tension en fonction de la différence de charge appliquée (niveau d'eau dans les tuyaux)
- Enregistrement des mesures dans Excel et logiciel Hobo

- Courbe de calibration par régression et obtention des paramètres dU/dH et Intercept

6.2. Calibration Voltage - Temperature

- Préparation du dispositif comme dans la photo et placement du capteur dans la chambre climatique



- Réalisation de 3 cycles de température pour chaque capteur, avec différents dH appliquée
- Régression avec script R du dossier calibration_molonari

Points à prendre en compte

- Planification du plan de travail pour calibrer les capteurs, notamment pour les cycles de température qui prend assez de temps
- Organisations des enregistrements dans les fichiers correspondants et avec le nom approprié comme décrit dans le protocole

- Téléchargement du logiciel Hoboware dans l'ordinateur personnel.
- Lecture des données du data logger: 
- Enregistrement des nouvelles mesures: 
- Faire attention qu'une fois qu'on commence un nouvel enregistrement, on perd tous les donnés antérieurs.

III - UTILISATION DU CODE ET INTERPRÉTATION DU RÉSULTAT

Nous avons pu utiliser le code de traitement de données sur nos données comme expliqué précédemment puis utilisé le code de traitement de la corrélation :

1. Choix des Points Corrélés et Visualisation

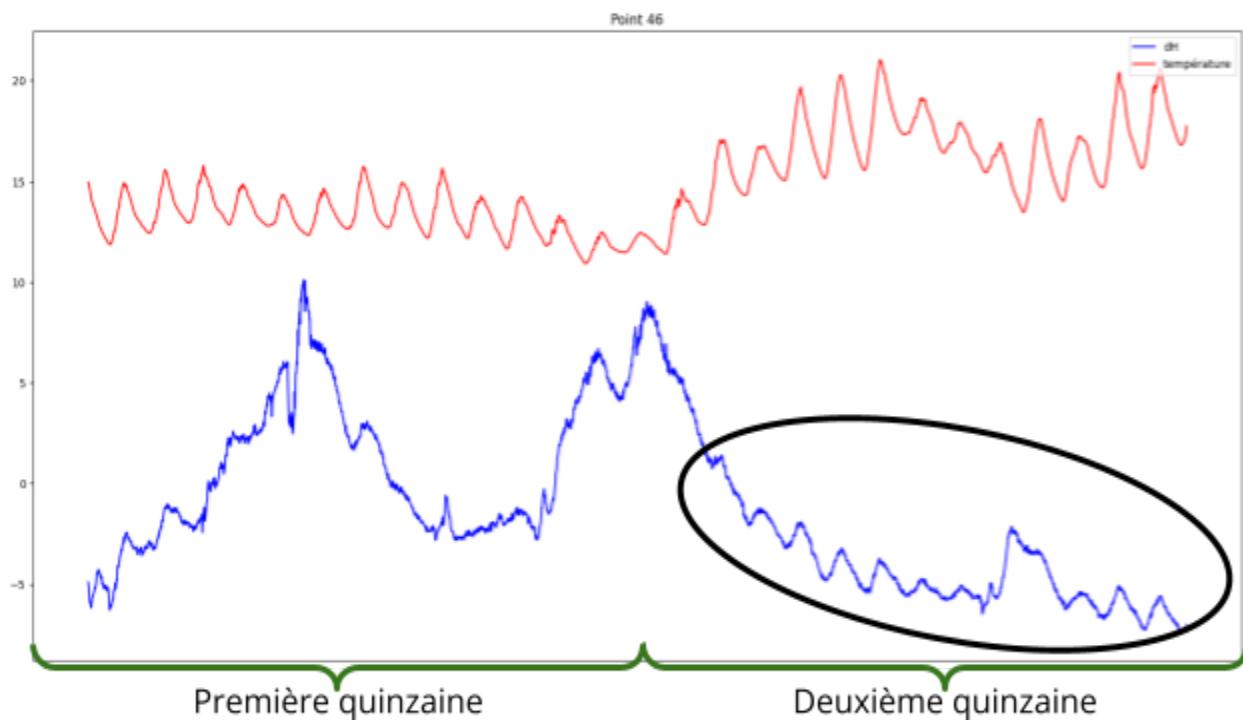
Affichage des courbes de dH et de température de la rivière afin de visualiser les corrélations entre les deux :

```
for i in range(len(numeroPoints)):
    plt.figure(figsize=(20,10))
    # on retire les valeurs de l'axe des abscisses pour accélerer l'affichage
    plt.xticks([])
    plt.plot(data[i]['pression2']['dates'], data[i]['pression2']['dH']*150, color = 'blue', label = 'dH')
    plt.plot(data[i]['pression2']['dates'], data[i]['pression2']['temperature_stream'], color = 'red', label = 'température')
    plt.title('Point ' + numeroPoints[i])
    plt.legend(loc='upper right')
    plt.show()
```

Méthode générale :

1. Affichage des courbes de dH et de température de la rivière par l'algorithme précédent. Observation d'apparition de sinusoïdes de même période que la périodicité journalière de la température.
2. Affichage du diagramme de corrélation (dH en fonction de la température rivière) sur l'intervalle de temps qui semble corrélé.
3. Observation d'apparition de linéarité et d'une pente de prédilection dans le diagramme de corrélation qui permet de justifier la corrélation entre la différence de charge et la température
4. Si point possède un intervalle corrélé: ajout du point dans la liste des points corrélés
5. Une fois dans la liste de points corrélés le point sera automatiquement traité par fft dans le code de décorrélation des données présenté précédemment

Cas pratique du point 46 ayant des relevées sur 30 jours pour décrire la démarche suivie :



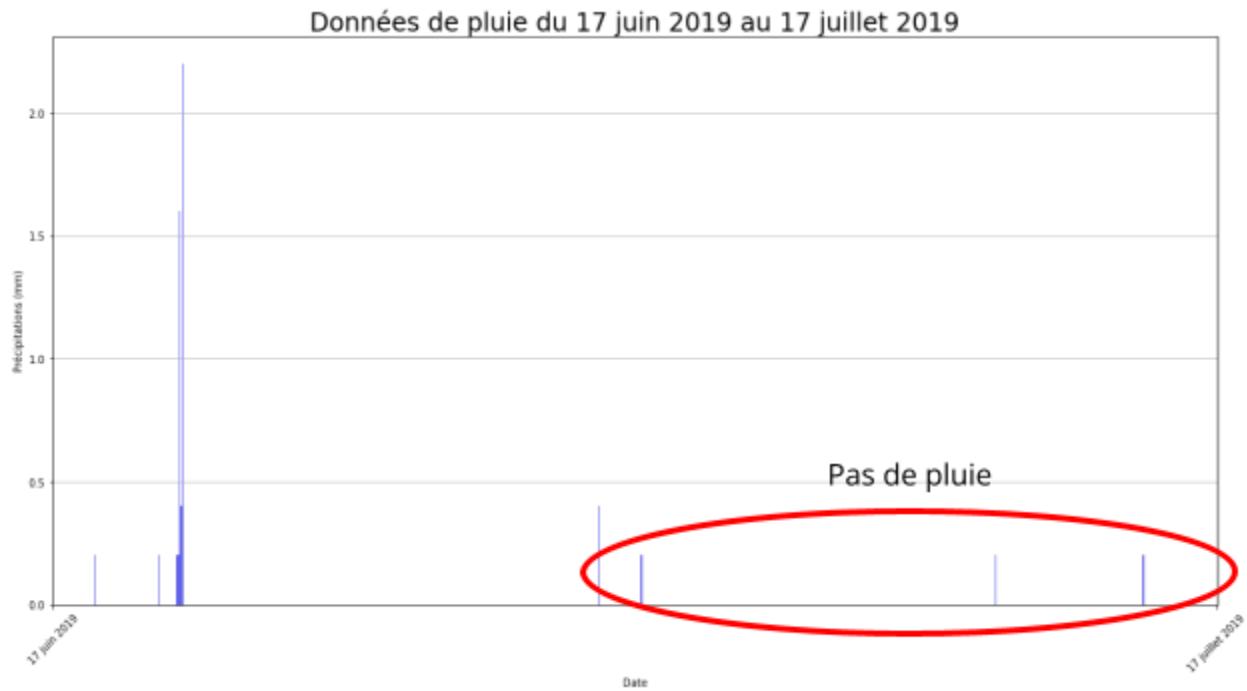
Observation de l'apparition au cours de la seconde quinzaine des relevées d'un bruit lié à la périodicité journalière de la température qui s'amplifie.

Raisons : nous n'avons pas pu déterminer les raisons exactes de l'apparition de cette corrélation mais nous avons fait des hypothèses. **A interpréter mieux l'année prochaine**

Hypothèses :

- Influence de la température sur les composants qui se dégradent au cours de l'implantation sur le terrain avec l'humidité. Les composants, comme vu lors de l'étalonnage, sont sensibles aux évolutions de températures. L'influence de la température s'amplifie au cours du temps.
- Evapotranspiration de la rivière.
- Double entrée analogue dans le capteur qui entraîne une désynchronisation des données.

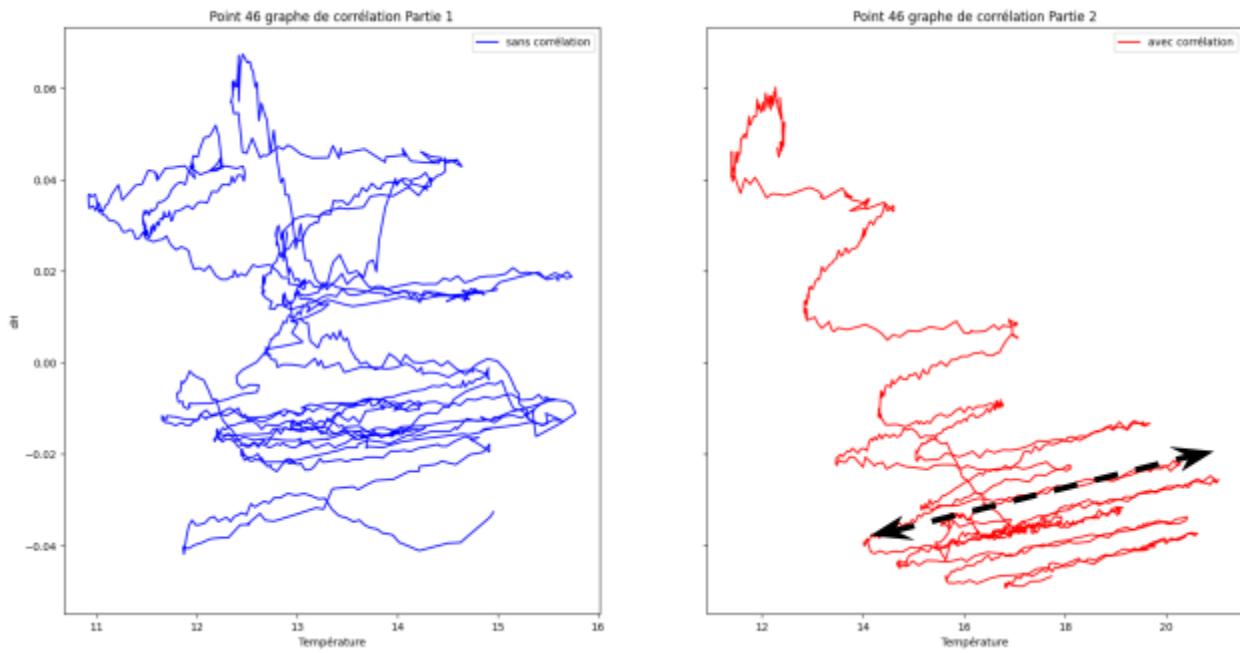
Cette périodicité n'est pas liée à une périodicité de la pluie car pour le point 46 il n'a pas plu sur la deuxième quinzaine des relevées (1 juillet 2017 - 17 juillet 2017) dans le bassin (Cf graphique suivant).



Ce graphique permet de prouver que la corrélation n'est pas dû à la pluie mais plus probablement à la température de la rivière qui varie fortement entre le jour et le soir pendant sur la deuxième quinzaine des relevées.

Vérification par diagramme de corrélation :

Une deuxième vérification est effectuée à travers un diagramme de corrélation divisé en deux parties. La première partie non corrélée est le graphe de corrélation des premiers quinze jours de relevés du point 46. La deuxième partie qui est corrélée est le graphe de corrélation de la deuxième quinzaine de relevés du point 46. La linéarité observée avec une pente caractéristique en pointillée justifie la corrélation.



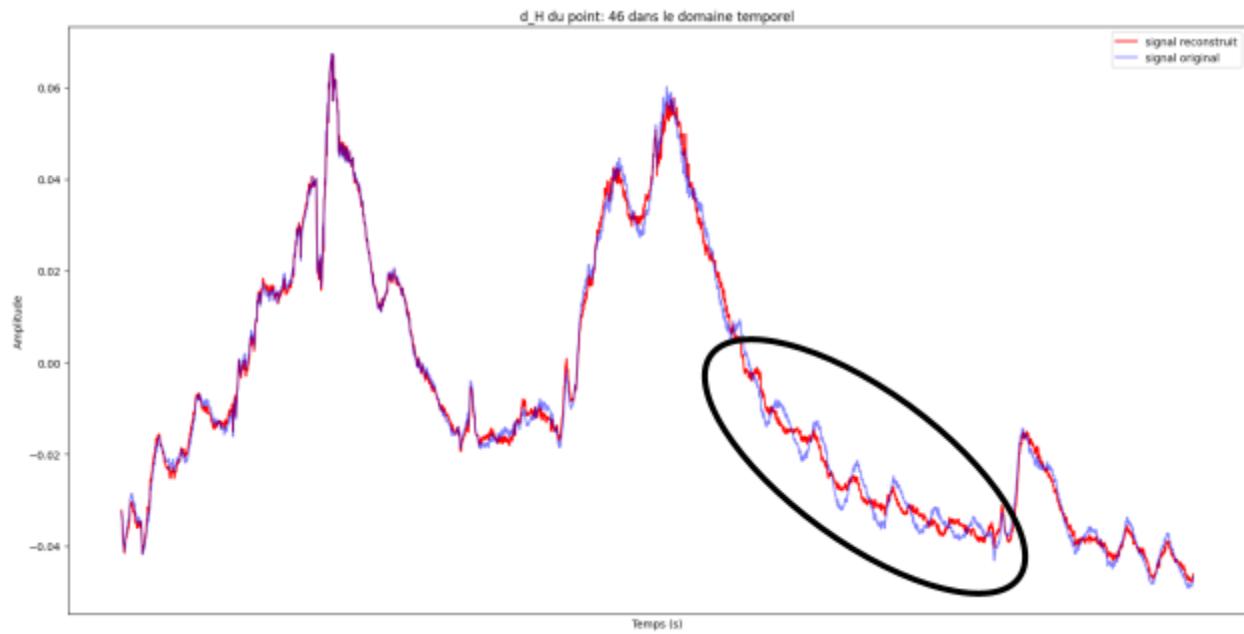
À l'aide de cette méthode, une liste des points corrélés est établie:

```
# Selection des points à traiter par fft (des points corrélés)
point_correler=['48','46','13','47','35']
```

2. Résultat de la méthode FFT

La méthode FFT est ensuite appliquée aux points identifiés comme corrélés. Le code de traitement FFT supprime la périodicité liée à la température. Par exemple, pour le point 46, la sinusoïde observée pendant la deuxième quinzaine de relevés est atténuée.

Exemple point 46 :

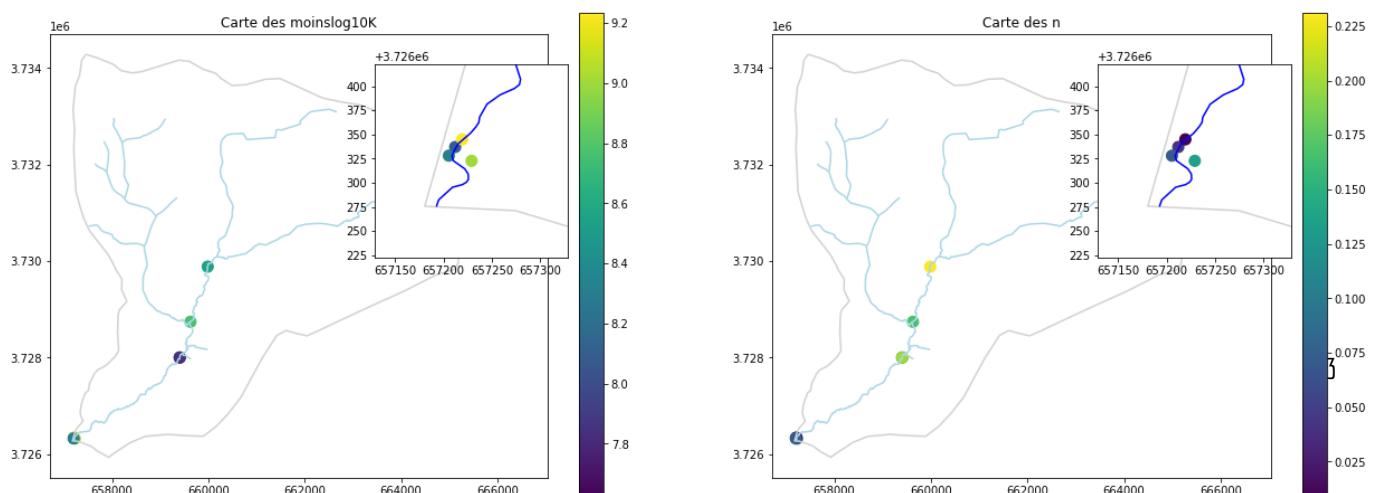


Cette approche permet de décorréliser les données de pression des effets indésirables liés à la température, améliorant ainsi la précision de l'analyse des variations de la charge hydraulique.

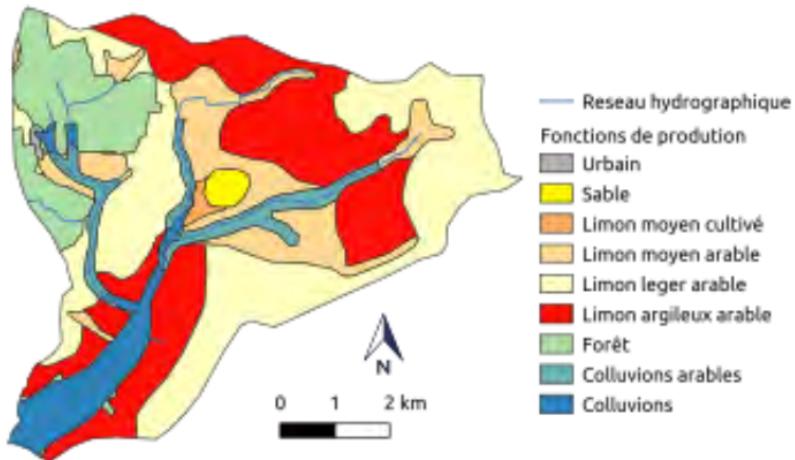
3. Interprétation des résultats

Nous avons effectué l'inversion de nos données puis tracé les cartes de nos paramètres sur les points : 13,14,35,36,48 et enfin 52,53, les points que nous avons récupérés cette année. Nous avons enregistré dans [params.csv](#) les meilleurs paramètres pour chaque point inversé pour tracer la carte.

Les deux cartes suivantes représentent l'évolution des paramètres de porosité et de perméabilité intrinsèque dans le bassin de l'Orgeval.

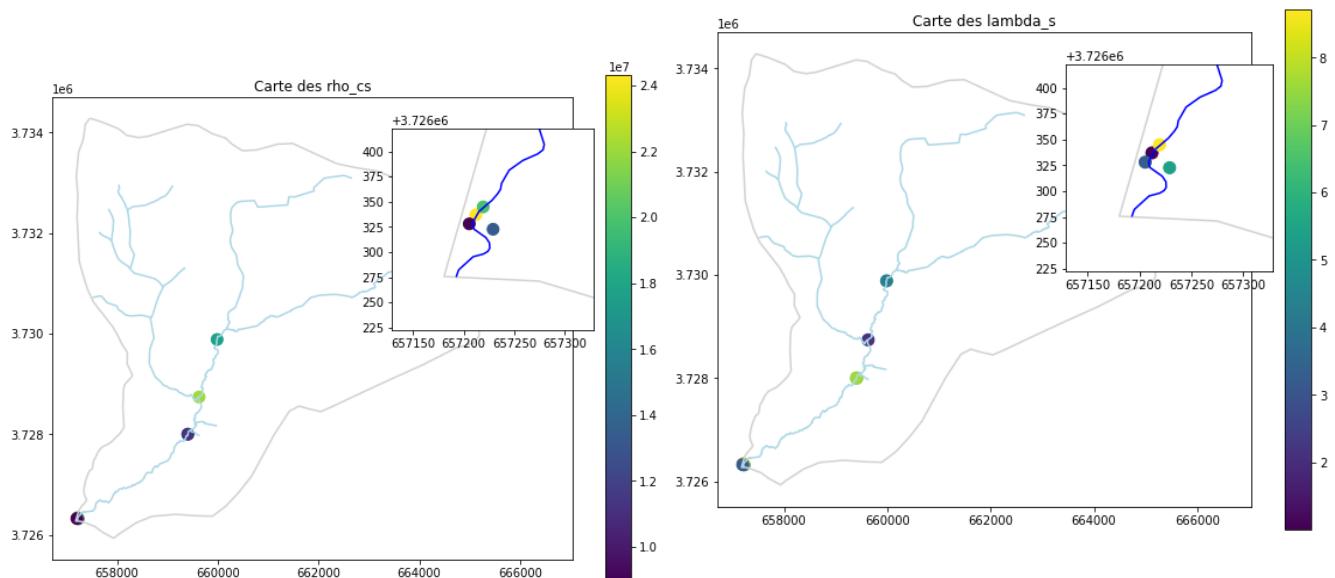


On peut remarquer que la porosité (n) augmente au nord du bassin. Cela est logique au vu de la composition des sols plus riche en limons et argiles au nord du bassin.

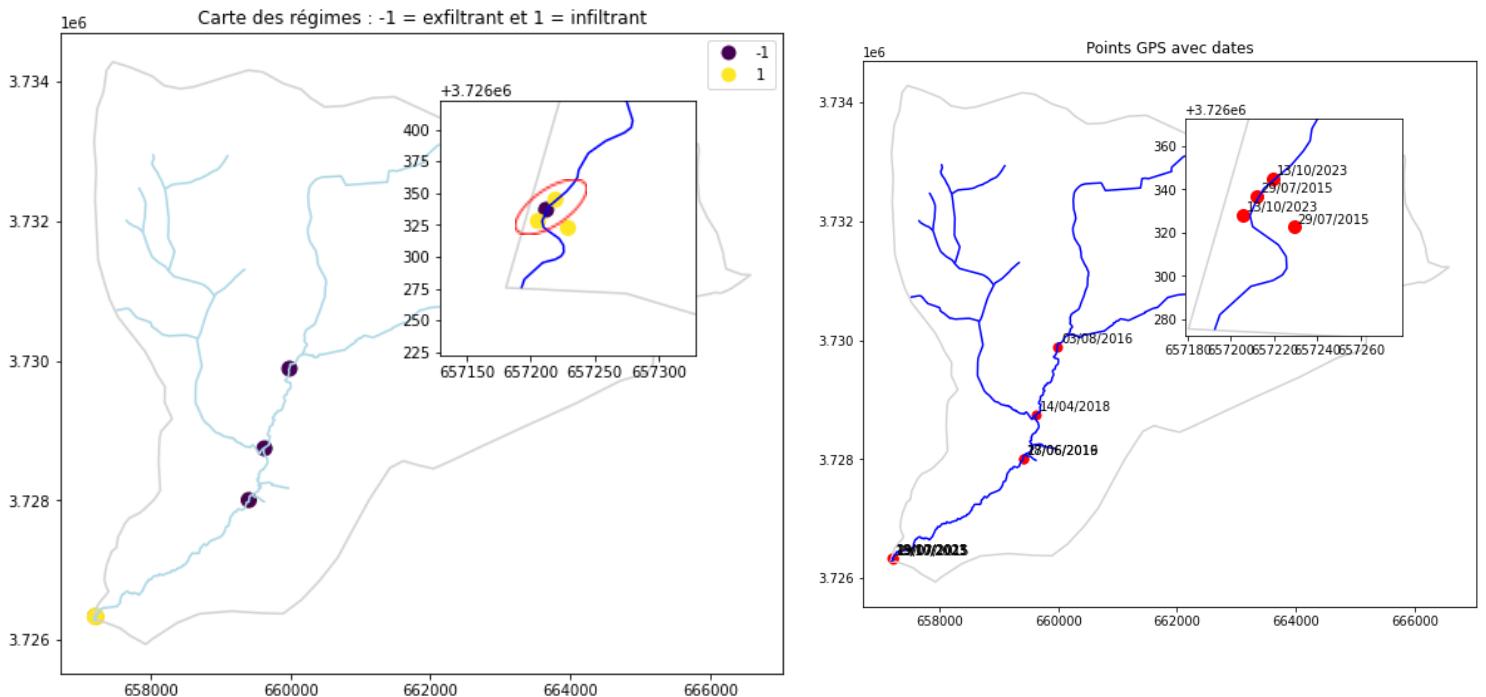


Rapport de synthèse PIREN-Seine (2019) Vol 4 Chap 3- CAttachment WAtter Quality Simulator : CaWaQS

On remarque de même que la perméabilité (moins $\log_{10} K$) est plus faible en amont. On a cependant une grande variabilité au niveau plus local qui semble corroborer des observations qui ont déjà été recensée dans la littérature (cf *Rapport de synthèse PIREN-Seine (2019) Vol 4 Chap 5- Variation spatiale des échanges nappe-rivière à l'échelle locale*).



Ces cartes montrent également les paramètres de capacité calorifique volumique et la conductivité thermique. On voit que la capacité semble globalement uniforme sur le bassin.



Enfin on peut tracer la carte du régime majoritaire pour chaque série spatio-temporelle de relevés. Il faut les comparer aux dates de ces relevés pour en faire une interprétation intéressante. Par exemple on remarque que sur les trois points au ru les avenelles (encerclés en rouge sur la carte) le régime est infiltrant pour des points effectués en octobre et exfiltrant pour des points effectués en juillet ce qui est logique. Les pluies d'octobre alimentent la nappe alors que la nappe alimente la rivière en plein été.