Tuto analyse fréquentielle de données sismiques avec Python sur JupyterLab (tuto_4_stations.ipynb)

1) Importer les librairies nécessaires

[1]:
```
import obspy
from obspy import read
from obspy.taup import TauPyModel
from obspy.signal.trigger import recursive_sta_lta, trigger_onset

import numpy as np

import matplotlib
import matplotlib.pyplot as plt
from matplotlib.pyplot import text

from math import acos, sin, cos, sqrt, atan2, radians

from scipy.fft import fft, fftfreq
```

2) Importer les données des 4 stations

Plus d'infos séisme de Grèce: https://www.emsc-csem.org/Earthquake/earthquake.php?id=954327

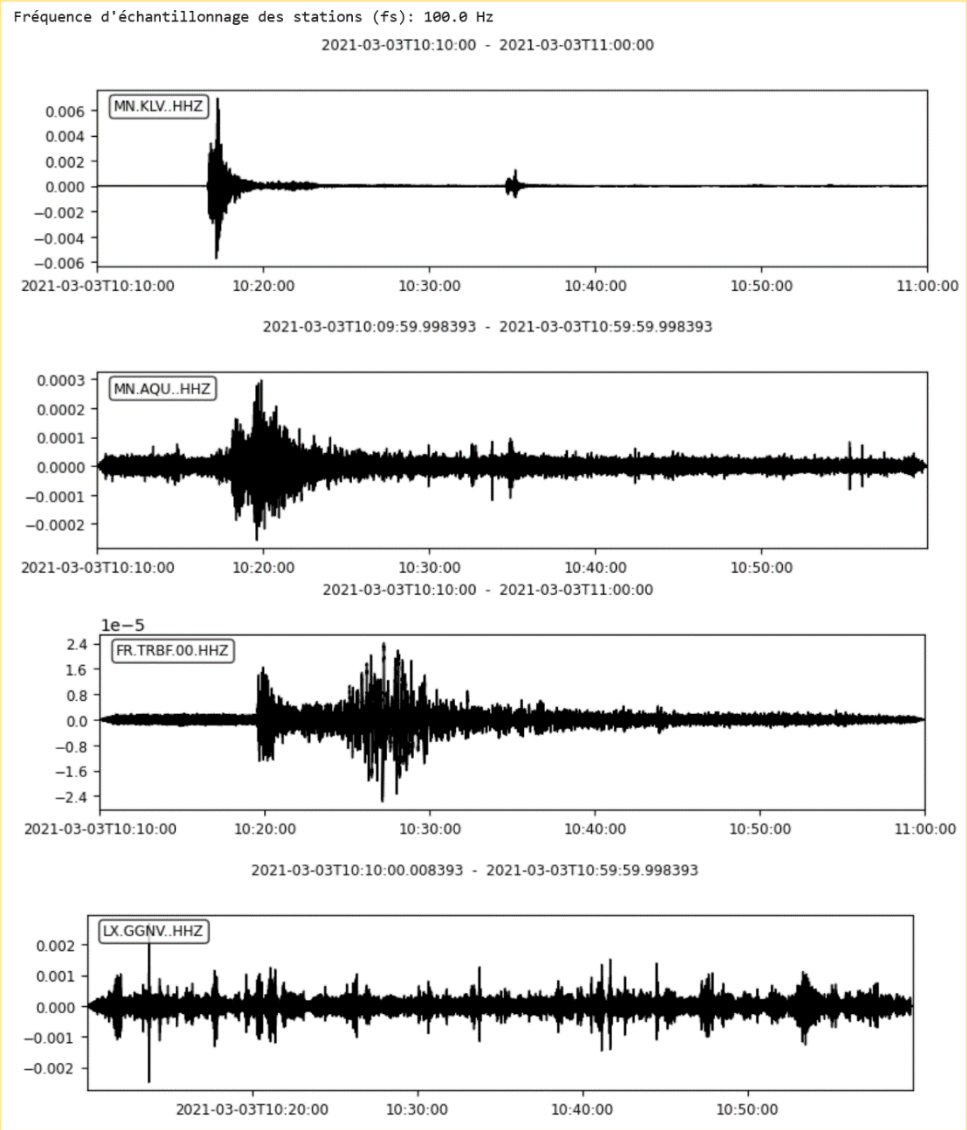
*[2]:
```
# Importer les données des différentes stations
# Séisme localisé en Grèce, de magnitude Mw=6.3, localisation: 39.76 N ; 22.21 E, profondeur = 8 km
# Même séisme enregistré par des stations de différents pays: Grèce (GR), Italie (IT), France (FR), Portugal (PO)

st_GR = read(r"votre_chemin\MN.KLV.HHZ.sac")
print(st_GR)

st_IT = read(r"votre_chemin\MN.AQU.HHZ.sac")
print(st_IT)

st_FR = read(r"votre_chemin\FR.TRBF.HHZ.sac")
print(st_FR)

st_PO = read(r"votre_chemin\LX.GGNV.HHZ.sac")
print(st_PO)
```


```
1 Trace(s) in Stream:
MN.KLV..HHZ | 2021-03-03T10:10:00.000000Z - 2021-03-03T11:00:00.000000Z | 100.0 Hz, 300001 samples
1 Trace(s) in Stream:
MN.AQU..HHZ | 2021-03-03T10:09:59.998393Z - 2021-03-03T10:59:59.998393Z | 100.0 Hz, 300001 samples
1 Trace(s) in Stream:
FR.TRBF.00.HHZ | 2021-03-03T10:10:00.000000Z - 2021-03-03T11:00:00.000000Z | 100.0 Hz, 300001 samples
1 Trace(s) in Stream:
LX.GGNV..HHZ | 2021-03-03T10:10:00.008393Z - 2021-03-03T10:59:59.998393Z | 100.0 Hz, 300000 samples
```

3) Premier aperçu des données

[3]:
```
# Informations sur les données:
print("Fréquence d'échantillonnage des stations (fs): " + str(st_GR[0].stats.sampling_rate) + " Hz")

# Afficher le contenu des données:
st_GR[0].plot()
plt.show()

st_IT[0].plot()
plt.show()

st_FR[0].plot()
plt.show()

st_PO[0].plot()
plt.show()
```

Shift + entrée



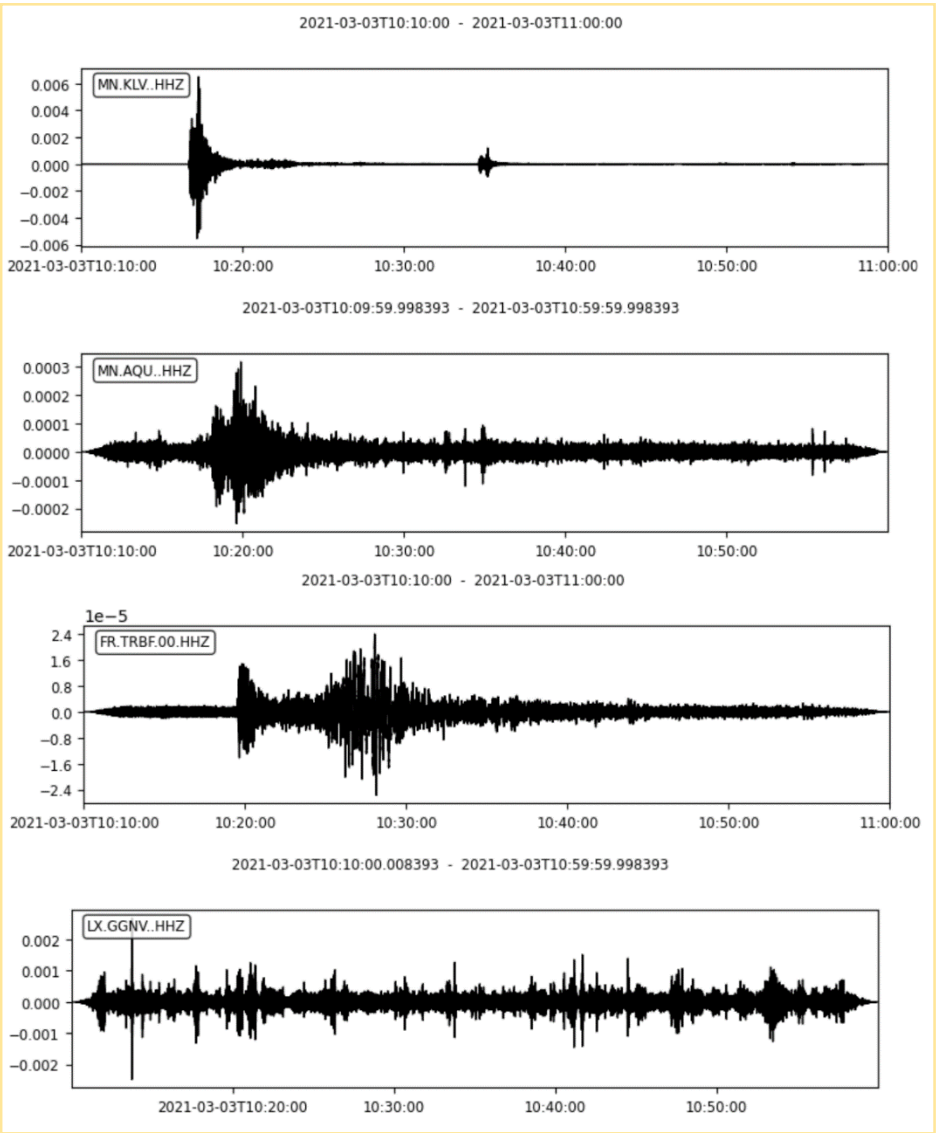
4) Pré-processing

detrend (polynomial): enlever la tendance du signal, ici de type polynomiale d'ordre 3
detrend (demean): enlever la moyenne du signal
taper: application d'un taper (atténuation des amplitudes aux 2 extrémités sur signal): peut fausser l'analyse fréquentielle
filter: filtrage des très basses fréquences du signal: élimination des fréquences d'origine naturelles (longues et petites oscillations)

[5]:
```
# Pré-traitement des données + affichage:
# Grèce
st_GR_corr = st_GR.copy()
st_GR_corr.detrend(type='polynomial', order=3)
st_GR_corr.detrend('demean')
st_GR_corr.taper(max_percentage=0.05, type="hann")
st_GR_corr.filter('highpass', freq=0.05)

st_GR_corr[0].plot()
plt.show()

# Italie
st_IT_corr = st_IT.copy()
st_IT_corr.detrend(type='polynomial', order=3)
st_IT_corr.detrend('demean')
st_IT_corr.taper(max_percentage=0.05, type="hann")
st_IT_corr.filter('highpass', freq=0.05)

st_IT_corr[0].plot()
plt.show()
```

```
# France
st_FR_corr = st_FR.copy()
st_FR_corr.detrend(type='polynomial', order=3)
st_FR_corr.detrend('demean')
st_FR_corr.taper(max_percentage=0.05, type="hann")
st_FR_corr.filter('highpass', freq=0.05)

st_FR_corr[0].plot()
plt.show()

# Portugal
st_PO_corr = st_PO.copy()
st_PO_corr.detrend(type='polynomial', order=3)
st_PO_corr.detrend('demean')
st_PO_corr.taper(max_percentage=0.05, type="hann")
st_PO_corr.filter('highpass', freq=0.05)

st_PO_corr[0].plot()
plt.show()
```

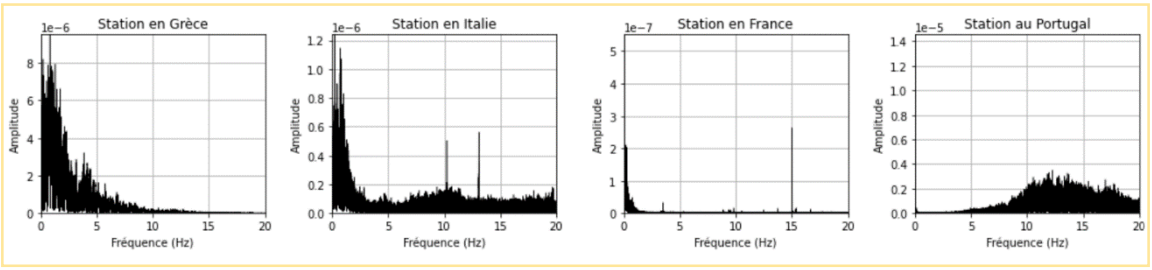
Shift + entrée



5) Calcul du spectre de Fourier (fréquences d'intérêt sismologique entre 0.5 et 20 Hz)

[6]:
```
# Calcul des spectres de Fourier:
fs = st_GR[0].stats.sampling_rate
dt = 1.0 / fs

# Nombre d'échantillons (de points) dans les sismogrammes
N_GR = st_GR_corr[0].stats.npts
N_IT = st_IT_corr[0].stats.npts
N_FR = st_FR_corr[0].stats.npts
N_PO = st_PO_corr[0].stats.npts

# Calcul de la transformée de Fourier (fast fourier transform)
FFT_GR = fft(st_GR_corr[0].data)
FFT_IT = fft(st_IT_corr[0].data)
FFT_FR = fft(st_FR_corr[0].data)
FFT_PO = fft(st_PO_corr[0].data)

# On garde uniquement les fréquences positives
# Valeurs en abscisses: xf
xf_GR = fftfreq(N_GR, dt)[:N_GR//2]
xf_IT = fftfreq(N_IT, dt)[:N_IT//2]
xf_FR = fftfreq(N_FR, dt)[:N_FR//2]
xf_PO = fftfreq(N_PO, dt)[:N_PO//2]

# On prend la valeur absolue de l'amplitude uniquement pour les fréquences positives et normalisation
# Valeurs en ordonnées: yf
yf_GR = np.abs(FFT_GR[0:N_GR//2]) * 2.0/N_GR
yf_IT = np.abs(FFT_IT[0:N_IT//2]) * 2.0/N_IT
yf_FR = np.abs(FFT_FR[0:N_FR//2]) * 2.0/N_FR
yf_PO = np.abs(FFT_PO[0:N_PO//2]) * 2.0/N_PO

# Afficher les spectres de Fourier des 3 séismes:
fig0, axs = plt.subplots(1, 4, figsize=(18, 3))
plt.subplots_adjust(wspace=0.3)
axs[0].plot(xf_GR, yf_GR, linewidth=0.5, color='black')
axs[0].set_ylim(0, max(yf_GR))
axs[0].set_title("Station en Grèce")
axs[1].plot(xf_IT, yf_IT, linewidth=0.5, color='black')
axs[1].set_ylim(0, max(yf_IT))
axs[1].set_title("Station en Italie")
axs[2].plot(xf_FR, yf_FR, linewidth=0.5, color='black')
axs[2].set_ylim(0, max(yf_FR))
axs[2].set_title("Station en France")
axs[3].plot(xf_PO, yf_PO, linewidth=0.5, color='black')
axs[3].set_ylim(0, max(yf_PO))
axs[3].set_title("Station au Portugal")

for num in range(4):
    axs[num].grid(True)
    axs[num].set_xlabel("Fréquence (Hz)")
    axs[num].set_ylabel("Amplitude")
    axs[num].set_xlim(0, 20)       # Pour mieux analyser le contenu fréquentiel (avec set_xlim):
                                   #        -> essayez de recentrer les graphiques entre [0-20] Hz, [0-5] Hz et [0-1] Hz

# Enregistrer la figure
#fig0.savefig(r"spectres_fourier.png", dpi=350, bbox_inches="tight")
```

Shift + entrée



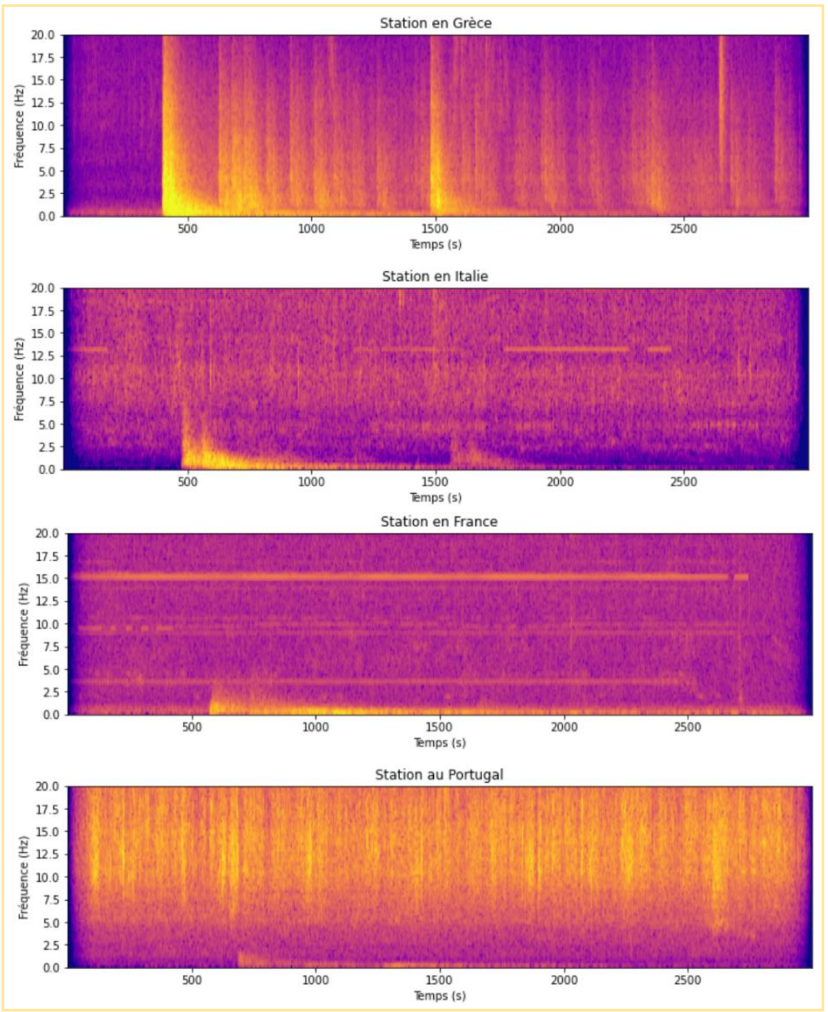
6) Figure spectrogramme

Paramètres du spectrogramme avec specgram:

- **NFFT:** nombre d'échantillons utilisés dans chaque bloc pour la FFT
- **Fs:** fréquence d'échantillonnage
- **window:** type de fenêtre utilisée (par défaut: hanning window) d'une longueure égale à celle du signal
- **noverlap:** nombre d'échantillons similaires repris pour le calculs de la FFT de bloc à bloc
- **cmap:** code couleur du spectrogramme
- **scale_by_freq:** donne les valeurs de densité (couleurs) en Hz^-1
- **vmin et vmax:** valeures minimales et maximales de l'amplitude du spectre (intensité min et max des couleurs)

[7]:
```
# Spectrogrammes
nfft = 400
ylim = 20

# Grèce
plt.figure(figsize=(12, 3))
plt.specgram(st_GR_corr[0], NFFT=nfft, Fs=fs, window=None, noverlap=None, cmap='plasma', scale_by_freq=True, vmin=-190, vmax=-70)
plt.ylim(0, ylim)
plt.margins(0,0)
plt.xlabel('Temps (s)')
plt.ylabel('Fréquence (Hz)')
plt.title('Station en Grèce')
plt.show()

# Italie
plt.figure(figsize=(12, 3))
plt.specgram(st_IT_corr[0], NFFT=nfft, Fs=fs, window=None, noverlap=None, cmap='plasma', scale_by_freq=True, vmin=-150, vmax=-80)
plt.ylim(0, ylim)
plt.margins(0,0)
plt.xlabel('Temps (s)')
plt.ylabel('Fréquence (Hz)')
plt.title('Station en Italie')
plt.show()

# France
plt.figure(figsize=(12, 3))
plt.specgram(st_FR_corr[0], NFFT=nfft, Fs=fs, window=None, noverlap=None, cmap='plasma', scale_by_freq=True, vmin=-190, vmax=-90)
plt.ylim(0, ylim)
plt.margins(0,0)
plt.xlabel('Temps (s)')
plt.ylabel('Fréquence (Hz)')
plt.title('Station en France')
plt.show()

# Portugal
plt.figure(figsize=(12, 3))
plt.specgram(st_PO_corr[0], NFFT=nfft, Fs=fs, window=None, noverlap=None, cmap='plasma', scale_by_freq=True, vmin=-160, vmax=-70)
plt.ylim(0, ylim)
plt.margins(0,0)
plt.xlabel('Temps (s)')
plt.ylabel('Fréquence (Hz)')
plt.title('Station au Portugal')
plt.show()
```

Shift + entrée



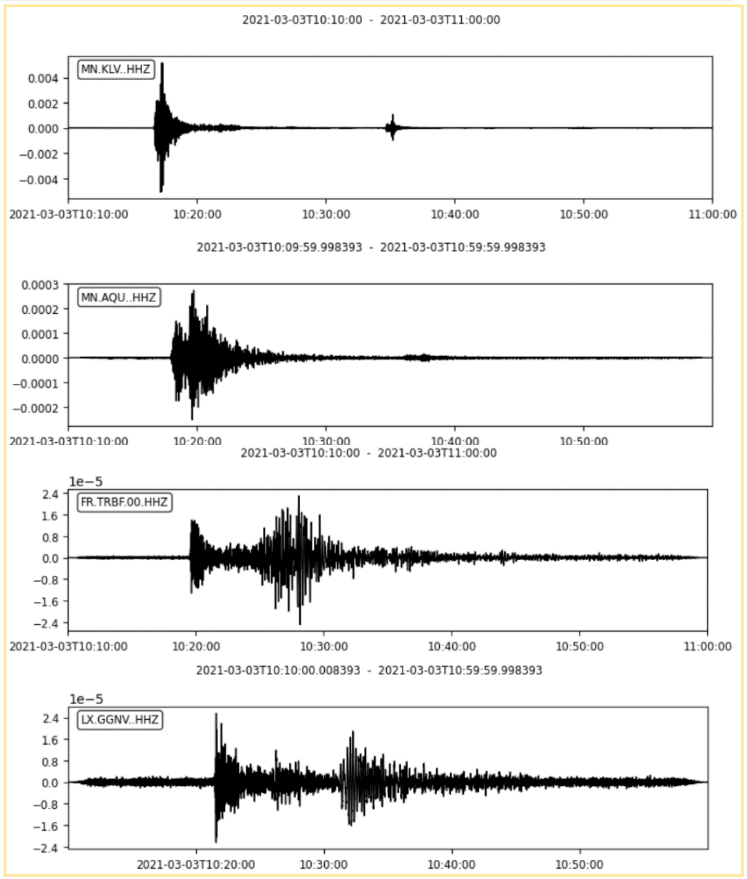
7) Rendre visibles les ondes qui nous intéressent

Après avoir identifié la gamme de fréquences du séisme sur le spectrogramme, nous pouvons filtrer le signal pour rendre visible les différentes ondes (P, S, de surface, etc).
Différents filtres sont possibles :

- **filtre 'highpass' ou 'passe haut':** permet de garder les fréquences supérieures à une fréquence donnée (seules les fréquences inférieures sont filtrées)
- **filtre 'lowpass' ou 'passe bas':** permet de garder les fréquences inférieures à une fréquence donnée (seules les fréquences supérieures sont filtrées)
- **filtre 'bandpass' ou 'passe bande':** permet de garder les fréquences se situant dans un intervalle donné (entre des fréquences minimale et maximale)

[8]:
```
# Filtrer le signal pour faire ressortir les fréquences qui nous intéressent:
# 'highpass' avec freq = ?
# 'lowpass' avec freq = ?
# 'bandpass' avec freqmin = ? et freqmax = ?

# Grèce
st_GR_filt = st_GR_corr.copy()
st_GR_filt.filter('lowpass', freq=2.5)

st_GR_filt[0].plot()
plt.show()

# Italie
st_IT_filt = st_IT_corr.copy()
st_IT_filt.filter('lowpass', freq=2.5)

st_IT_filt[0].plot()
plt.show()

# France
st_FR_filt = st_FR_corr.copy()
st_FR_filt.filter('lowpass', freq=1.5)

st_FR_filt[0].plot()
plt.show()

# Portugal
st_PO_filt = st_PO_corr.copy()
st_PO_filt.filter('lowpass', freq=1)

st_PO_filt[0].plot()
plt.show()
```

Shift + entrée



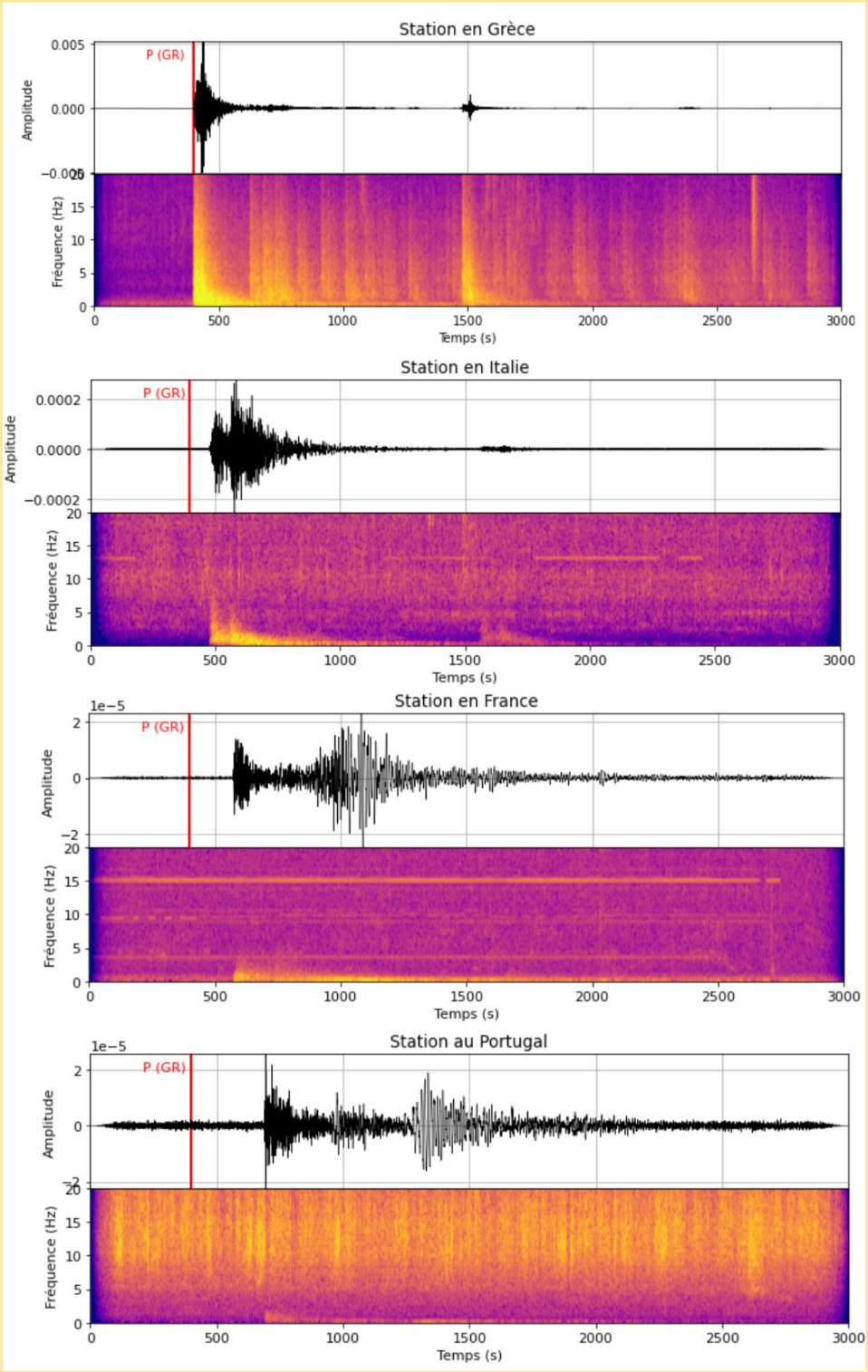
8) Pointer la première arrivée de l'onde P sur la station en Grèce (la plus proche du séisme)

[9]:
```
# Pointer les arrivées approximatives des ondes sur la station en Grèce (ne pas modifier les paramètres)
cft = recursive_sta_lta(st_GR_corr[0].data, int(2.5 * fs), int(10. * fs))
on_of = trigger_onset(cft, 3.5, 0.5)
```

9) Figure groupée : sismogramme et spectrogramme

[10]:
```
# Figures groupées: sismogramme et spectrogramme
# Vecteur de temps du signal
tt_GR = np.linspace(0, dt * N_GR, N_GR)
tt_IT = np.linspace(0, dt * N_IT, N_IT)
tt_FR = np.linspace(0, dt * N_FR, N_FR)
tt_PO = np.linspace(0, dt * N_PO, N_PO)

# Grèce
fig1, axs = plt.subplots(2, sharex=True, sharey=False, figsize=(11,4))
plt.subplots_adjust(wspace=0, hspace=0)
axs[0].plot(tt_GR, st_GR_filt[0].data, color='black', linewidth=0.5)
axs[0].vlines(on_of[0, 0]*(1/fs), min(st_GR_filt[0].data), max(st_GR_filt[0].data), color='r', linewidth=2)
text(210, 39, "P (GR)", verticalalignment='top', color='r')
axs[0].grid()
axs[0].margins(0, 0)
axs[0].set_ylabel('Amplitude')
axs[1].specgram(st_GR_corr[0], NFFT=nfft, Fs=fs, window=None, noverlap=None, cmap='plasma', scale_by_freq=True, vmin=-190, vmax=-70)
axs[1].margins(0, 0)
axs[1].set_ylim(0, 20)
axs[1].set_xlabel('Temps (s)')
axs[1].set_ylabel('Fréquence (Hz)')
matplotlib.rcParams.update({'font.size': 11})
axs[0].set_title("Station en Grèce")

# Italie
fig2, axs = plt.subplots(2, sharex=True, sharey=False, figsize=(11,4))
plt.subplots_adjust(wspace=0, hspace=0)
axs[0].plot(tt_IT, st_IT_filt[0].data, color='black', linewidth=0.5)
axs[0].vlines(on_of[0, 0]*(1/fs), min(st_IT_filt[0].data), max(st_IT_filt[0].data), color='r', linewidth=2)
text(210, 39, "P (GR)", verticalalignment='top', color='r')
axs[0].grid()
axs[0].margins(0, 0)
axs[0].set_ylabel('Amplitude')
axs[1].specgram(st_IT_corr[0], NFFT=nfft, Fs=fs, window=None, noverlap=None, cmap='plasma', scale_by_freq=True, vmin=-150, vmax=-80)
axs[1].margins(0, 0)
axs[1].set_ylim(0, 20)
axs[1].set_xlabel('Temps (s)')
axs[1].set_ylabel('Fréquence (Hz)')
matplotlib.rcParams.update({'font.size': 11})
axs[0].set_title("Station en Italie")

# France
fig3, axs = plt.subplots(2, sharex=True, sharey=False, figsize=(11,4))
plt.subplots_adjust(wspace=0, hspace=0)
axs[0].plot(tt_FR, st_FR_filt[0].data, color='black', linewidth=0.5)
axs[0].vlines(on_of[0, 0]*(1/fs), min(st_FR_filt[0].data), max(st_FR_filt[0].data), color='r', linewidth=2)
text(210, 39, "P (GR)", verticalalignment='top', color='r')
axs[0].grid()
axs[0].margins(0, 0)
axs[0].set_ylabel('Amplitude')
axs[1].specgram(st_FR_corr[0], NFFT=nfft, Fs=fs, window=None, noverlap=None, cmap='plasma', scale_by_freq=True, vmin=-190, vmax=-90)
axs[1].margins(0, 0)
axs[1].set_ylim(0, 20)
axs[1].set_xlabel('Temps (s)')
axs[1].set_ylabel('Fréquence (Hz)')
matplotlib.rcParams.update({'font.size': 11})
axs[0].set_title("Station en France")

# Portugal
fig4, axs = plt.subplots(2, sharex=True, sharey=False, figsize=(11,4))
plt.subplots_adjust(wspace=0, hspace=0)
axs[0].plot(tt_PO, st_PO_filt[0].data, color='black', linewidth=0.5)
axs[0].vlines(on_of[0, 0]*(1/fs), min(st_PO_filt[0].data), max(st_PO_filt[0].data), color='r', linewidth=2)
text(210, 39, "P (GR)", verticalalignment='top', color='r')
axs[0].grid()
axs[0].margins(0, 0)
axs[0].set_ylabel('Amplitude')
axs[1].specgram(st_PO_corr[0], NFFT=nfft, Fs=fs, window=None, noverlap=None, cmap='plasma', scale_by_freq=True, vmin=-160, vmax=-70)
axs[1].margins(0, 0)
axs[1].set_ylim(0, 20)
axs[1].set_xlabel('Temps (s)')
axs[1].set_ylabel('Fréquence (Hz)')
matplotlib.rcParams.update({'font.size': 11})
axs[0].set_title("Station au Portugal")
```

Shift + entrée



[ ]:
```
# Enregistrer les figures
fig1.savefig(r"station_grece.png", dpi=350, bbox_inches="tight")
fig2.savefig(r"station_italie.png", dpi=350, bbox_inches="tight")
fig3.savefig(r"station_france.png", dpi=350, bbox_inches="tight")
fig4.savefig(r"station_portugal.png", dpi=350, bbox_inches="tight")
```

10) Quels types d'ondes arrivent à jusqu'aux stations ?

[11]:
```
# Quelles sont les différentes ondes détectables ?
# Station en Grèce: lat=38.04367° lon=22.15042°
# Station en Italie: lat=42.354° lon=13.405°
# Station en France: lat=44.1054° lon=3.9587°
# Station au Portugal: lat=38.7885° lon=-9.1505

# Calcul de la distance en degrés rad:
# latitude et longitude de l'événement sismique (à ne pas changer)
lat_event = radians(39.76)
lon_event = radians(22.21)

# latitude et longitude de la station
lat_station=radians(38.718498)
lon_station=radians(-9.1505)

Rplanet=6378.14
dlat=lat_station-lat_event
dlon=lon_station-lon_event

a = (sin(dlat/2))**2 + cos(lat_station) * cos(lat_event) * (sin(dlon/2))**2
c = 2 * atan2(sqrt(a), sqrt(1-a))

distkm=c*Rplanet
distrad=c
```

```
# Calcul des temps d'arrivées théoriques:
model = TauPyModel(model="iasp91")
arrivals = model.get_travel_times(source_depth_in_km=8, distance_in_degree=distrad)
print(arrivals)
```

Shift + entrée

```
17 arrivals
        p phase arrival at 8.205 seconds
        sP phase arrival at 10.033 seconds
        s phase arrival at 14.163 seconds
        PcP phase arrival at 509.897 seconds
        ScP phase arrival at 721.046 seconds
        PcS phase arrival at 722.047 seconds
        ScS phase arrival at 933.199 seconds
        PKiKP phase arrival at 993.191 seconds
        pPKiKP phase arrival at 995.949 seconds
        sPKiKP phase arrival at 996.951 seconds
        SKiKP phase arrival at 1204.337 seconds
        PKIKKIKP phase arrival at 1911.521 seconds
        SKIKKIKP phase arrival at 2122.668 seconds
        PKIKKIKS phase arrival at 2123.669 seconds
        SKIKKIKS phase arrival at 2334.816 seconds
        PKIKPPKIKP phase arrival at 2422.788 seconds
        SKIKSSKIKS phase arrival at 3270.380 seconds
```

11) Afficher les trajets des différentes ondes théoriques :

[12]:
```
# Afficher le trajet des ondes entre le séisme et la station
arrivals = model.get_ray_paths(source_depth_in_km=8, distance_in_degree=distrad, phase_list=["p", "PcP", "PKiKP"])
ax = arrivals.plot_rays(plot_type="cartesian")
```

Shift + entrée