DUACS-L4

Analyse des données DUACS/L4

Entrée [1]:

```
%matplotlib inline

import numpy as np
import xarray as xr
import matplotlib.pyplot as plt
import dask.array as da

plt.rcParams['figure.figsize'] = (15,10)
```

Initialisation de l'ensemble de données

Ici, nous chargeons l'ensemble des données depuis un dépôt `zarr`. Notez que ce très grand ensemble de données s'initialise presque instantanément, et nous pouvons voir la liste complète des variables et des coordonnées.

Entrée [3]:

```
dirname = "/work/ALT/odatis/AVISO/dataset-duacs-rep-global-merged-allsat-phy-l4/"

ds = xr.open_zarr(dirname)
ds
```

Out[3]:

```
<xarray.Dataset>
Dimensions:    (latitude: 720, longitude: 1440, nv: 2, time: 9629)
Coordinates:
  * latitude   (latitude) float64 -89.88 -89.62 -89.38 ... 89.38 89.62 89.88
  * longitude  (longitude) float64 0.125 0.375 0.625 0.875 ... 359.4 359.6 3
59.9
  * nv         (nv) int32 0 1
  * time       (time) datetime64[ns] 1993-01-01 1993-01-02 ... 2019-05-13
Data variables:
    adt        (time, latitude, longitude) float64 dask.array<chunksize=(32,
720, 1440), meta=np.ndarray>
    crs        int32 ...
    err        (time, latitude, longitude) float64 dask.array<chunksize=(32,
720, 1440), meta=np.ndarray>
    lat_bnds   (latitude, nv) float64 dask.array<chunksize=(720, 2), meta=n
p.ndarray>
    lon_bnds   (longitude, nv) float64 dask.array<chunksize=(1440, 2), meta=
np.ndarray>
    sla        (time, latitude, longitude) float64 dask.array<chunksize=(32,
720, 1440), meta=np.ndarray>
    ugos       (time, latitude, longitude) float64 dask.array<chunksize=(32,
720, 1440), meta=np.ndarray>
    ugosa      (time, latitude, longitude) float64 dask.array<chunksize=(32,
720, 1440), meta=np.ndarray>
    vgos       (time, latitude, longitude) float64 dask.array<chunksize=(32,
720, 1440), meta=np.ndarray>
    vgosa      (time, latitude, longitude) float64 dask.array<chunksize=(32,
720, 1440), meta=np.ndarray>
Attributes:
    Conventions:                     CF-1.6
    Metadata_Conventions:            Unidata Dataset Discovery v1.0
    cdm_data_type:                   Grid
    comment:                         Sea Surface Height measured by Altimetr
y...
    contact:                         servicedesk.cmems@mercator-ocean.eu
    creator_email:                   servicedesk.cmems@mercator-ocean.eu
    creator_name:                    CMEMS - Sea Level Thematic Assembly Cen
ter
    creator_url:                     http://marine.copernicus.eu
    geospatial_lat_max:              89.875
    geospatial_lat_min:              -89.875
    geospatial_lat_resolution:       0.25
    geospatial_lat_units:            degrees_north
    geospatial_lon_max:              359.875
```

```
    geospatial_lon_min:             0.125
    geospatial_lon_resolution:      0.25
    geospatial_lon_units:           degrees_east
    geospatial_vertical_max:        0.0
    geospatial_vertical_min:        0.0
    geospatial_vertical_positive:   down
    geospatial_vertical_resolution: point
    geospatial_vertical_units:      m
    institution:                    CLS, CNES
    keywords:                       Oceans > Ocean Topography > Sea Surface
...
    keywords_vocabulary:            NetCDF COARDS Climate and Forecast Stan
d...
    license:                        http://marine.copernicus.eu/web/27-serv
i...
    platform:                       ERS-1, Topex/Poseidon,
    processing_level:               L4
    product_version:                2019
    project:                        COPERNICUS MARINE ENVIRONMENT MONITORIN
G...
    references:                     http://marine.copernicus.eu
    software_version:               6.2_DUACS_DT2018_baseline
    source:                         Altimetry measurements
    ssalto_duacs_comment:           The reference mission used for the alti
m...
    standard_name_vocabulary:       NetCDF Climate and Forecast (CF) Metada
t...
    summary:                        SSALTO/DUACS Delayed-Time Level-4 sea s
u...
    title:                          DT merged all satellites Global Ocean G
r...
```

Examiner visuellement certaines des données

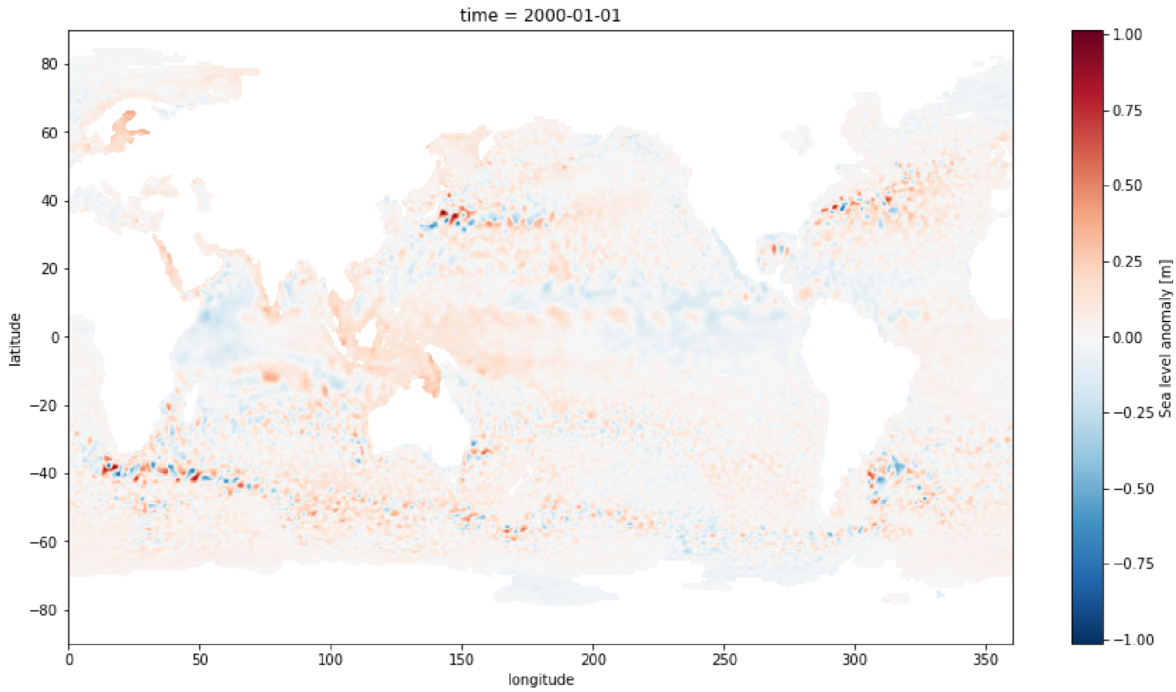
Vérifions que les données manipulées sont correctes:

Entrée [4]:

```
plt.rcParams['figure.figsize'] = (15, 8)
ds.sla.sel(time='2000-01-01', method='nearest').plot()
```

Out[4]:

```
<matplotlib.collections.QuadMesh at 0x2ae7ba14d6a0>
```



Creation d'un cluster sur le HPC

Entrée [6]:

```
import dask_jobqueue


cluster = dask_jobqueue.PBSCluster(
    cores=2,
    processes=1,
    memory="4GB",
    interface="ib0",
    project='duacs_l4',
    walltime='04:00:00',
    local_directory='$TMPDIR')
cluster.adapt(minimum=1, maximum=10)
cluster
```

```
/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/distributed/
node.py:151: UserWarning: Port 8787 is already in use.
Perhaps you already have a cluster running?
Hosting the HTTP server on port 34400 instead
  warnings.warn(
```

```
VBox(children=(HTML(value='<h2>PBSCluster</h2>'), HBox(children=(HTML(value
='\n<div>\n  <style scoped>\n    .d…
```

Entrée [7]:

```
import dask.distributed


client = dask.distributed.Client(cluster)
client
```

Out[7]:

Client	Cluster
Scheduler: tcp://10.135.39.36:42318	**Workers:** 1
Dashboard: http://10.135.39.36:34400/status (http://10.135.39.36:34400/status)	**Cores:** 2
	Memory: 4.00 GB

Série chronologique du niveau moyen de la mer

Nous faisons ici un calcul simple mais fondamental : le taux d'augmentation du niveau moyen de la mer sur la période d'observation.

Entrée [8]:

```
# La taille des données à traiter
ds.sla.nbytes/1e9
```

Out[8]:

```
79.8667776
```

Entrée [9]:

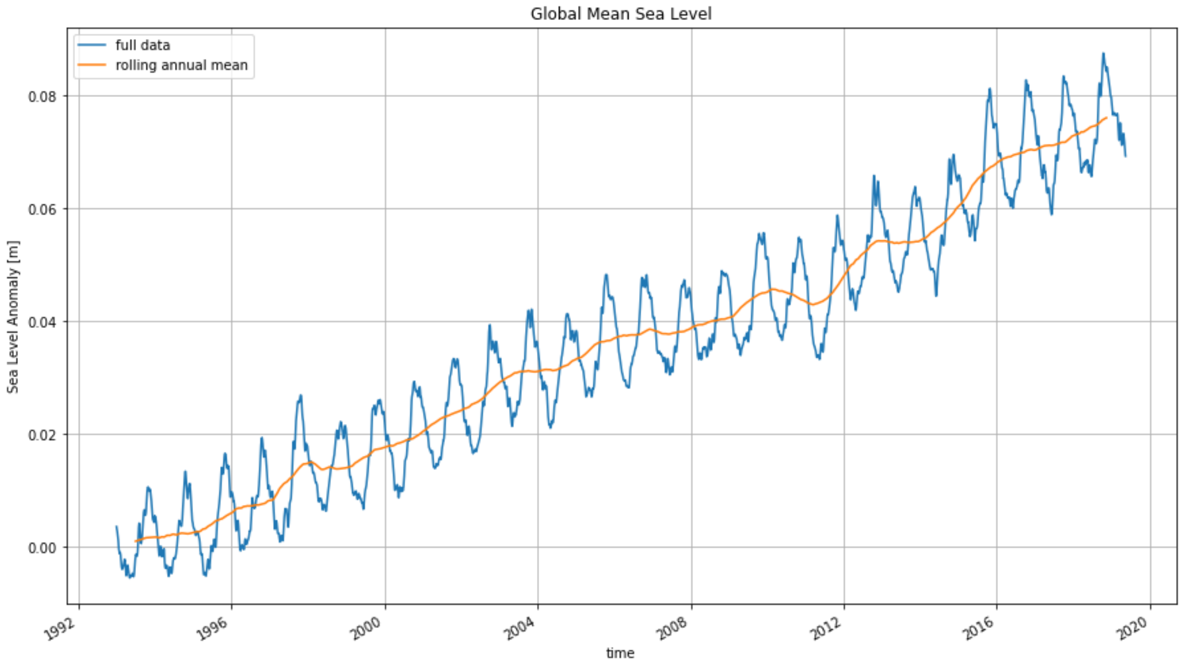
```
%%time

# L'étape de calcul intensif
sla_timeseries = ds.sla.mean(dim=('latitude', 'longitude')).load()
```

```
CPU times: user 4.02 s, sys: 478 ms, total: 4.5 s
Wall time: 41 s
```

Entrée [10]:

```
sla_timeseries.plot(label='full data')
sla_timeseries.rolling(time=365, center=True).mean().plot(label='rolling annual mean')
plt.ylabel('Sea Level Anomaly [m]')
plt.title('Global Mean Sea Level')
plt.legend()
plt.grid()
```



Afin de comprendre comment l'élévation du niveau de la mer est répartie en latitude, nous pouvons faire une sorte de [diagramme de Hovmöller (https://en.wikipedia.org/wiki/Hovm%C3%B6ller_diagram)](https://en.wikipedia.org/wiki/Hovm%C3%B6ller_diagram).

Entrée [11]:

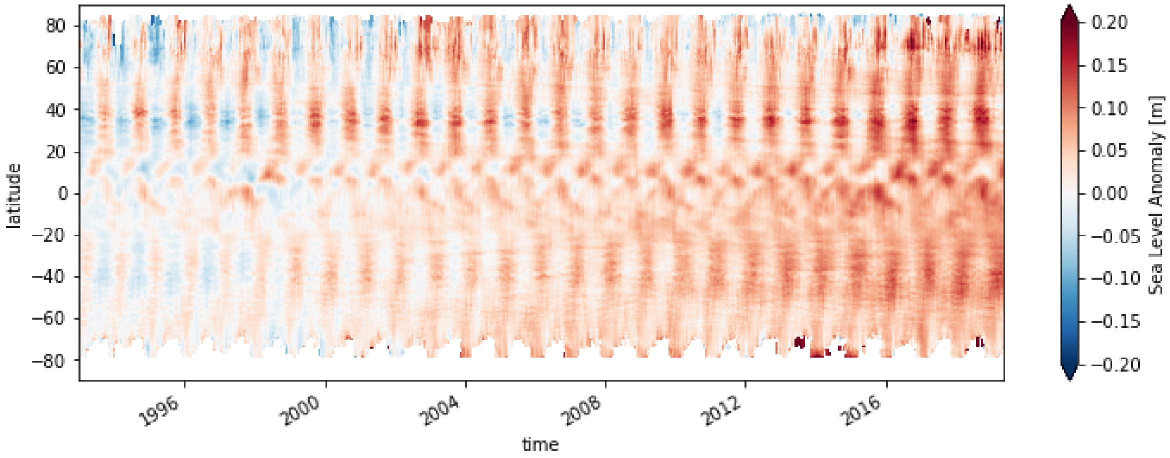
```
sla_hov = ds.sla.mean(dim='longitude').load()
```

Entrée [12]:

```
fig, ax = plt.subplots(figsize=(12, 4))
sla_hov.name = 'Sea Level Anomaly [m]'
sla_hov.transpose().plot(vmax=0.2, ax=ax)
```

Out[12]:

```
<matplotlib.collections.QuadMesh at 0x2ae7db560c10>
```



Nous pouvons voir que la plus grande partie de l'élévation du niveau de la mer se situe en fait dans l'hémisphère sud.

Variabilité du niveau de la mer

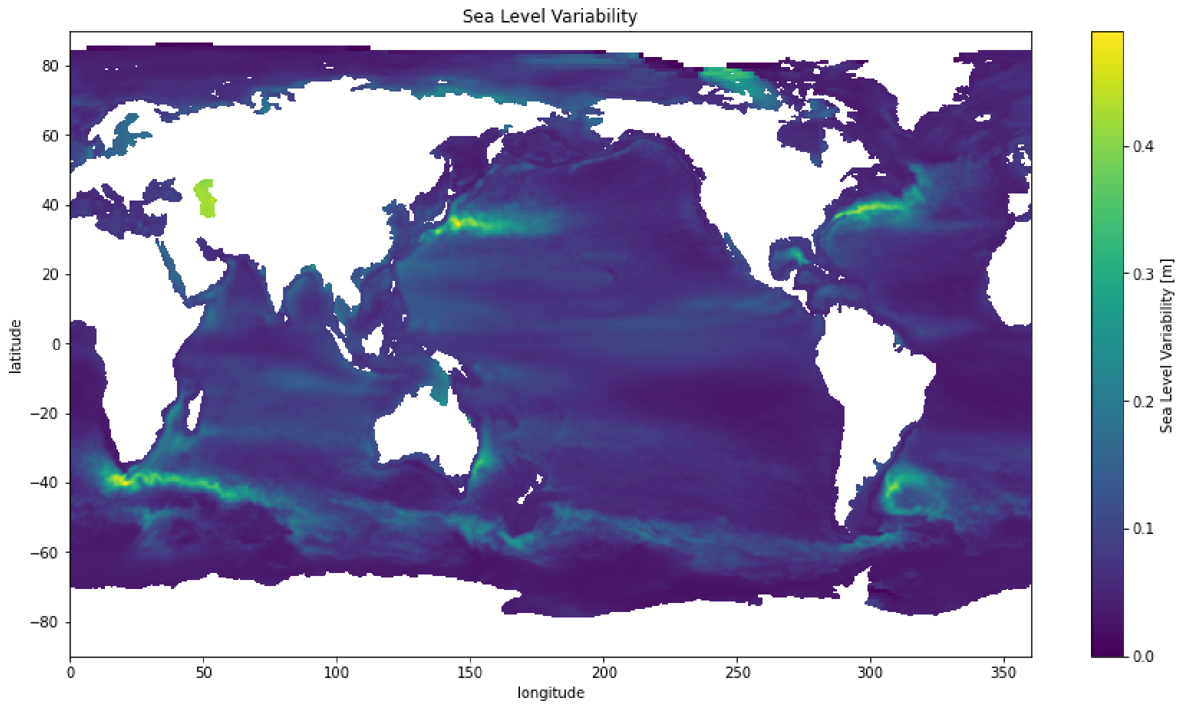
Nous pouvons examiner la variabilité naturelle du niveau de la mer en regardant son écart-type dans le temps.

Entrée [13]:

```
sla_std = ds.sla.std(dim='time').load()
sla_std.name = 'Sea Level Variability [m]'
```

Entrée [14]:

```
ax = sla_std.plot()
_ = plt.title('Sea Level Variability')
```



Analyse spectrale

Nous effectuons ici une analyse spectrale de la fréquence du nombre d'ondes du signal de SSH en utilisant des méthodes similaires à celles décrites dans [Abernathey & Wortham (2015) (https://journals.ametsoc.org/doi/10.1175/JPO-D-14-0160.1)](https://journals.ametsoc.org/doi/10.1175/JPO-D-14-0160.1).

Étape 1 : Extraction d'un secteur dans le Pacifique Est

Ce secteur est choisi parce qu'il dispose de très peu de terres.

Entrée [15]:

```
sector = ds.sla.sel(longitude=slice(180, 230), latitude=slice(-70, 55, 4))
sector_anom = (sector - sector.mean(dim='longitude'))
sector_anom
```

Out[15]:

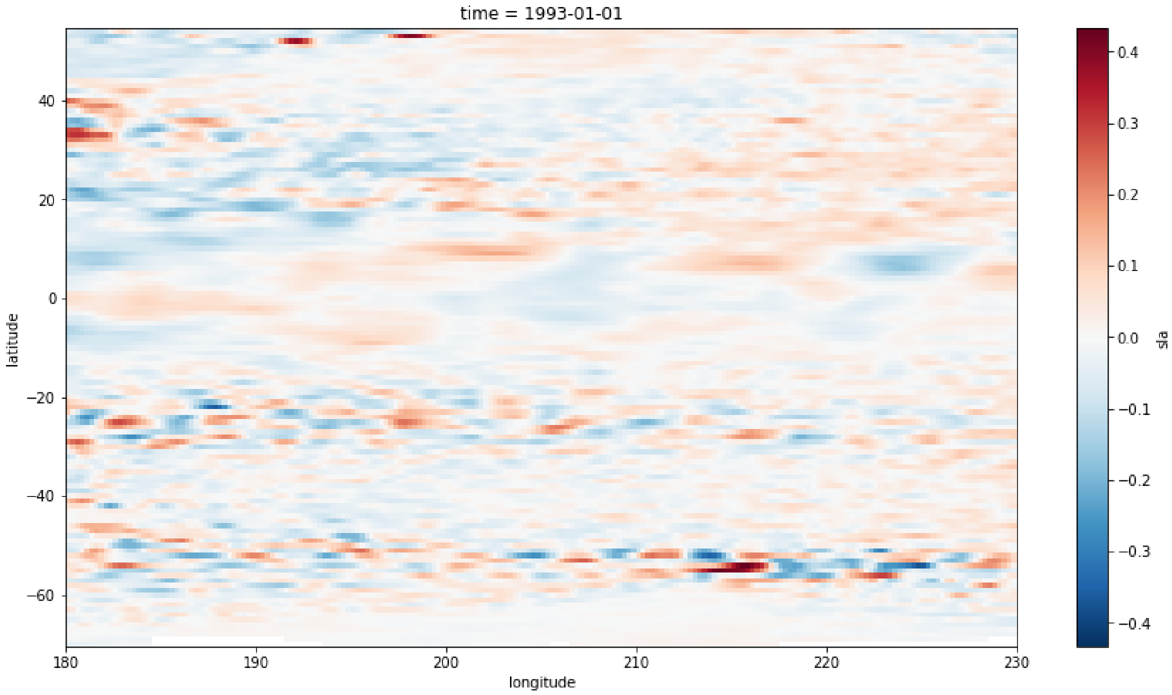
```
<xarray.DataArray 'sla' (time: 9629, latitude: 125, longitude: 200)>
dask.array<sub, shape=(9629, 125, 200), dtype=float64, chunksize=(32, 125, 2
00), chunktype=numpy.ndarray>
Coordinates:
  * latitude   (latitude) float64 -69.88 -68.88 -67.88 ... 52.12 53.12 54.12
  * longitude  (longitude) float64 180.1 180.4 180.6 180.9 ... 229.4 229.6 2
29.9
  * time       (time) datetime64[ns] 1993-01-01 1993-01-02 ... 2019-05-13
```

Entrée [16]:

```
sector_anom[0].plot()
```

Out[16]:

```
<matplotlib.collections.QuadMesh at 0x2ae800201340>
```



Étape 2: Reconstruire, remodeler et fenêtrer les données pour préparer efficacement le calcul de la FFT

Entrée [17]:

```
# remodeler les données en matrices de 365 jours et les réorganiser les tableaux en mémoire
nsegments = 24
segment_len = 365
sector_reshape = (sector_anom.isel(time=slice(0, nsegments*segment_len))
                             .transpose('latitude', 'time', 'longitude')
                             .chunk({'time': segment_len}))
sector_reshape
```

Out[17]:

```
<xarray.DataArray 'sla' (latitude: 125, time: 8760, longitude: 200)>
dask.array<rechunk-merge, shape=(125, 8760, 200), dtype=float64, chunksize=
(125, 365, 200), chunktype=numpy.ndarray>
Coordinates:
  * latitude   (latitude) float64 -69.88 -68.88 -67.88 ... 52.12 53.12 54.12
  * longitude  (longitude) float64 180.1 180.4 180.6 180.9 ... 229.4 229.6 2
29.9
  * time       (time) datetime64[ns] 1993-01-01 1993-01-02 ... 2016-12-25
```

Entrée [18]:

```
# maintenant un récupére le tableau dask
data = sector_reshape.data

arrays = [data[:, n*segment_len:(n + 1)*segment_len][np.newaxis]
          for n in range(nsegments)]
stacked = da.concatenate(arrays)
stacked
```

Out[18]:

	Array	Chunk
Bytes	1.75 GB	73.00 MB
Shape	(24, 125, 365, 200)	(1, 125, 365, 200)
Count	2798 Tasks	24 Chunks
Type	float64	numpy.ndarray

24 1 200 365 125

Entrée [19]:

```
# appliquer les fenêtres sur les données
data_windowed = (stacked
                 * np.hanning(stacked.shape[-1])[None, None, None, :]
                 * np.hanning(stacked.shape[-2])[None, None, :, None])
```

Étape 3: Calculer effectivement la transformée de Fourier et la densité spectrale de puissance

Entrée [20]:

```
# FFT
data_fft = da.fft.fftn(data_windowed, axes=(-2, -1))

# Spectre de puissance et moyenne sur les segments
power_spec = np.real(data_fft * np.conj(data_fft)).mean(axis=0)
power_spec
```

Out[20]:

	Array	Chunk
Bytes	73.00 MB	73.00 MB
Shape	(125, 365, 200)	(125, 365, 200)
Count	2977 Tasks	1 Chunks
Type	float64	numpy.ndarray

200 365 125

Entrée [21]:

```
# Calcul et charge les résultats en mémoire
power_spec_shift = np.fft.fftshift(power_spec.compute(), axes=(-2, -1))
```

```
distributed.utils - ERROR -
Traceback (most recent call last):
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/utils.py", line 655, in log_errors
    yield
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/scheduler.py", line 3515, in retire_workers
    await self.replicate(
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/scheduler.py", line 3274, in replicate
    assert count > 0
AssertionError
distributed.core - ERROR -
Traceback (most recent call last):
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/core.py", line 528, in handle_comm
    result = await result
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/scheduler.py", line 3515, in retire_workers
    await self.replicate(
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/scheduler.py", line 3274, in replicate
    assert count > 0
AssertionError
distributed.utils - ERROR -
Traceback (most recent call last):
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/utils.py", line 655, in log_errors
    yield
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/deploy/adaptive.py", line 187, in scale_down
    await self.scheduler.retire_workers(
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/core.py", line 812, in send_recv_from_rpc
    result = await send_recv(comm=comm, op=key, **kwargs)
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/core.py", line 682, in send_recv
    raise exc.with_traceback(tb)
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/core.py", line 528, in handle_comm
    result = await result
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/scheduler.py", line 3515, in retire_workers
    await self.replicate(
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/scheduler.py", line 3274, in replicate
    assert count > 0
AssertionError
tornado.application - ERROR - Exception in callback functools.partial(<bound
method IOLoop._discard_future_result of <zmq.eventloop.ioloop.ZMQIOLoop obje
ct at 0x2ae7bbca98b0>>, <Task finished name='Task-292378' coro=<AdaptiveCor
e.adapt() done, defined at /softs/rh7/conda-envs/pangeo_202012/lib/python3.
8/site-packages/distributed/deploy/adaptive_core.py:179> exception=Assertion
Error()>)
Traceback (most recent call last):
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/torn
```

```
ado/ioloop.py", line 741, in _run_callback
    ret = callback()
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/torn
ado/ioloop.py", line 765, in _discard_future_result
    future.result()
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/deploy/adaptive_core.py", line 203, in adapt
    await self.scale_down(**recommendations)
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/deploy/adaptive.py", line 187, in scale_down
    await self.scheduler.retire_workers(
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/core.py", line 812, in send_recv_from_rpc
    result = await send_recv(comm=comm, op=key, **kwargs)
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/core.py", line 682, in send_recv
    raise exc.with_traceback(tb)
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/core.py", line 528, in handle_comm
    result = await result
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/scheduler.py", line 3515, in retire_workers
    await self.replicate(
  File "/softs/rh7/conda-envs/pangeo_202012/lib/python3.8/site-packages/dist
ributed/scheduler.py", line 3274, in replicate
    assert count > 0
AssertionError
```

Étape 4: Définir les coordonnées spectrales et tout rassembler dans un DataArray

Entrée [22]:

```
freq = np.fft.fftshift(np.fft.fftfreq(segment_len))

# La longueur d'onde est un peu plus délicate car elle dépend de la latitude
R = 6.37e6
# en km
dx = np.deg2rad(0.25) * R * np.cos(np.deg2rad(sector.latitude)) / 1000
inv_wavelength = np.vstack([np.fft.fftshift(np.fft.fftfreq(len(sector.longitude), d))
                            for d in dx.values])

ps_da = xr.DataArray(power_spec_shift, dims=('latitude', 'freq', 'wavenumber'),
                     coords={'latitude': sector.latitude,
                             'freq': ('freq', -freq, {'units': r'days$^{-1}$'}),
                              'inverse_wavelength': (('latitude', 'wavenumber'),
                                                     inv_wavelength, {'units': r'km$^{-1}$'
                     name='SSH_power_spectral_density')
ps_da
```

Out[22]:

```
<xarray.DataArray 'SSH_power_spectral_density' (latitude: 125, freq: 365, wa
venumber: 200)>
array([[[            nan,             nan,             nan, ...,
                     nan,             nan,             nan],
        [            nan,             nan,             nan, ...,
                     nan,             nan,             nan],
        [            nan,             nan,             nan, ...,
                     nan,             nan,             nan],
        ...,
        [            nan,             nan,             nan, ...,
                     nan,             nan,             nan],
        [            nan,             nan,             nan, ...,
                     nan,             nan,             nan],
        [            nan,             nan,             nan, ...,
                     nan,             nan,             nan]],

       [[            nan,             nan,             nan, ...,
                     nan,             nan,             nan],
        [            nan,             nan,             nan, ...,
                     nan,             nan,             nan],
        [            nan,             nan,             nan, ...,
                     nan,             nan,             nan],
...
        [1.08398899e-05, 9.75617559e-06, 7.52946669e-06, ...,
         8.09454475e-06, 1.00422469e-05, 7.02225630e-06],
        [8.80102204e-06, 9.61390007e-06, 1.13834824e-05, ...,
         8.23621628e-06, 8.93558453e-06, 8.51600719e-06],
        [9.45209779e-06, 9.77775549e-06, 1.13536292e-05, ...,
         1.15640639e-05, 1.08063110e-05, 9.53551082e-06]],

       [[8.09582663e-06, 8.04442968e-06, 7.69118265e-06, ...,
         6.35786351e-06, 9.25052901e-06, 8.71575457e-06],
        [6.31084572e-06, 7.30641509e-06, 6.12726067e-06, ...,
         1.00510755e-05, 1.00625545e-05, 8.21150558e-06],
```

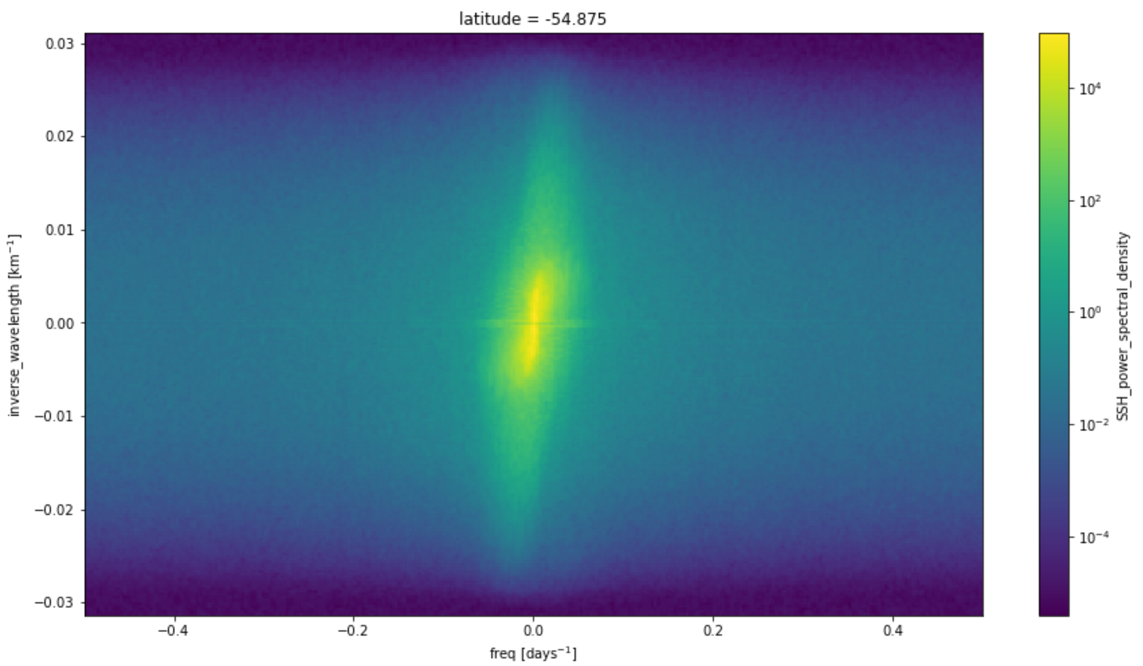
```
        [6.87443933e-06, 6.83419653e-06, 5.91478660e-06, ...,
         9.92557973e-06, 7.67243268e-06, 7.23983009e-06],
        ...,
        [6.87443933e-06, 7.23983009e-06, 7.67243268e-06, ...,
         7.11400271e-06, 5.91478660e-06, 6.83419653e-06],
        [6.31084572e-06, 8.21150558e-06, 1.00625545e-05, ...,
         6.12838223e-06, 6.12726067e-06, 7.30641509e-06],
        [8.09582663e-06, 8.71575457e-06, 9.25052901e-06, ...,
         5.63078087e-06, 7.69118265e-06, 8.04442968e-06]]])
Coordinates:
  * latitude            (latitude) float64 -69.88 -68.88 -67.88 ... 53.12 5
4.12
  * freq                (freq) float64 0.4986 0.4959 0.4932 ... -0.4959 -0.4
986
    inverse_wavelength  (latitude, wavenumber) float64 -0.05228 ... 0.03039
Dimensions without coordinates: wavenumber
```

Etape 5: Tracer les spectres de puissance à différentes latitudes

Entrée [23]:

```python
from matplotlib.colors import LogNorm

for lat in range(-55, 55, 10):
    plt.figure()
    (ps_da.sel(latitude=lat, method='nearest')
          .swap_dims({'wavenumber': 'inverse_wavelength'})
          .transpose().plot(norm=LogNorm()))
```



Après avoir passé en revue toute cette complexité, vous serez peut-être intéressé de savoir qu'il existe une bibliothèque qui facilite l'analyse spectrale des ensembles de données `xarray` :

- [https://xrft.readthedocs.io/en/latest/ (https://xrft.readthedocs.io/en/latest/)](https://xrft.readthedocs.io/en/latest/)

Avec xrft, nous aurions pu réduire toutes les étapes ci-dessus à quelques lignes de code. Mais nous n'en aurions pas appris autant ! 🤪

Entrée [ ]:

```
client.close()
cluster.close()
```

Entrée [ ]:

```
import os

for item in os.listdir("."):
    if item.startswith("dask-worker."):
        os.unlink(item)
```

Entrée [ ]:

```
```