In [ ]:

CoastSat.slope: Pekalongan

This is an extention of the main CoastSat toolbox and it is assumed that the user is familiar with CoastSat as the outputs of CoastSat are used here to estimate beach slopes. The `coastsat` environment also needs to be installed before attempting this example.

This example shows how to estimate the beach slope along transects in Pekalongan, Indonesia.

Initial settings

In [1]:

```
# initial settings
%load_ext autoreload
%autoreload 2
import os
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
import pytz
import pickle

# beach slope estmation module
import SDS_slope
```

1. Load satellite-derived shorelines and transect locations

Satellite-derived shorelines from Landsat 5, 7 and 8 between 1999 and 2020 are needed to estimate the beach slope, these have to be mapped with CoastSat beforehand. When mapping shorelines with CoastSat, the coordinates of the 2D shorelines are saved in a file named `sitename_output.pkl` .

In this example we use 2 files that are under *example_data/* (you will need the same files for another site):

- `NARRA_output.pkl` : satellite-derived shorelines mapped from 1999-2020 using Landsat 5,7 and 8 (no Sentinel-2)
- `NARRA_transects.geojson` : cross-shore transect coordinates (2 points, the first one being landwards)

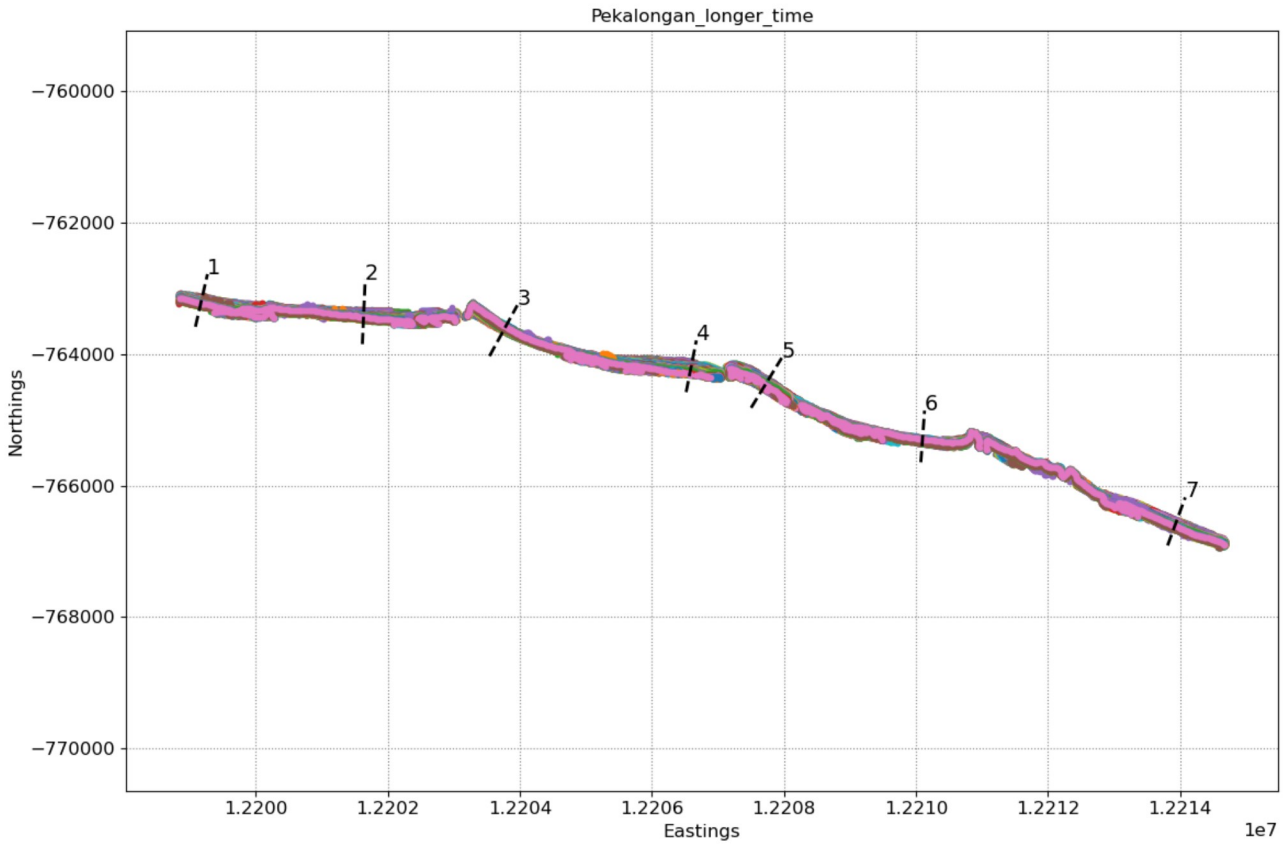
When preparing your own files, make sure that both files are in the same coordinate system (in this example epsg:28356).

The section below loads the two files, removes duplicates and shorelines with poor georeferencing and plots the 2D shorelines and cross-shore transects.

In [2]:

```
# load the sitename_output.pkl generated by CoastSat
sitename = 'Pekalongan_longer_time'
with open(os.path.join('data','Pekalongan_longer_time', sitename + '_output' + '.pkl'), 'rb') as f:
    output = pickle.load(f)

# load the 2D transects from geojson file
geojson_file = os.path.join(os.getcwd(), 'data','Pekalongan_longer_time', sitename + '_transects.geojson')
transects = SDS_slope.transects_from_geojson(geojson_file)

# remove S2 shorelines (the slope estimation algorithm needs only Landsat shorelines)
#if 'S2' in output['satname']:
#    idx_S2 = np.array([_ == 'S2' for _ in output['satname']])
#    for key in output.keys():
#        output[key] = [output[key][_] for _ in np.where(~idx_S2)[0]]

#print(len(output))

# remove duplicates (can happen that images overlap and there are 2 shorelines for the same date)
output = SDS_slope.remove_duplicates(output)
# remove shorelines from images with poor georeferencing (RMSE > 10 m)
output = SDS_slope.remove_inaccurate_georef(output, 10)

# plot shorelines and transects
fig,ax = plt.subplots(1,1,figsize=[12,  8])
fig.set_tight_layout(True)
ax.axis('equal')
ax.set(xlabel='Eastings', ylabel='Northings', title=sitename)
ax.grid(linestyle=':', color='0.5')
for i in range(len(output['shorelines'])):
    coords = output['shorelines'][i]
    date = output['dates'][i]
    ax.plot(coords[:,0], coords[:,1], '.', label=date.strftime('%d-%m-%Y'))
for key in transects.keys():
    ax.plot(transects[key][:,0],transects[key][:,1],'k--',lw=2)
    ax.text(transects[key][-1,0], transects[key][-1,1], key)
```

```
7 transects have been loaded
0 duplicates
0 bad georef
```



2. Extract time-series of shoreline change along the transects

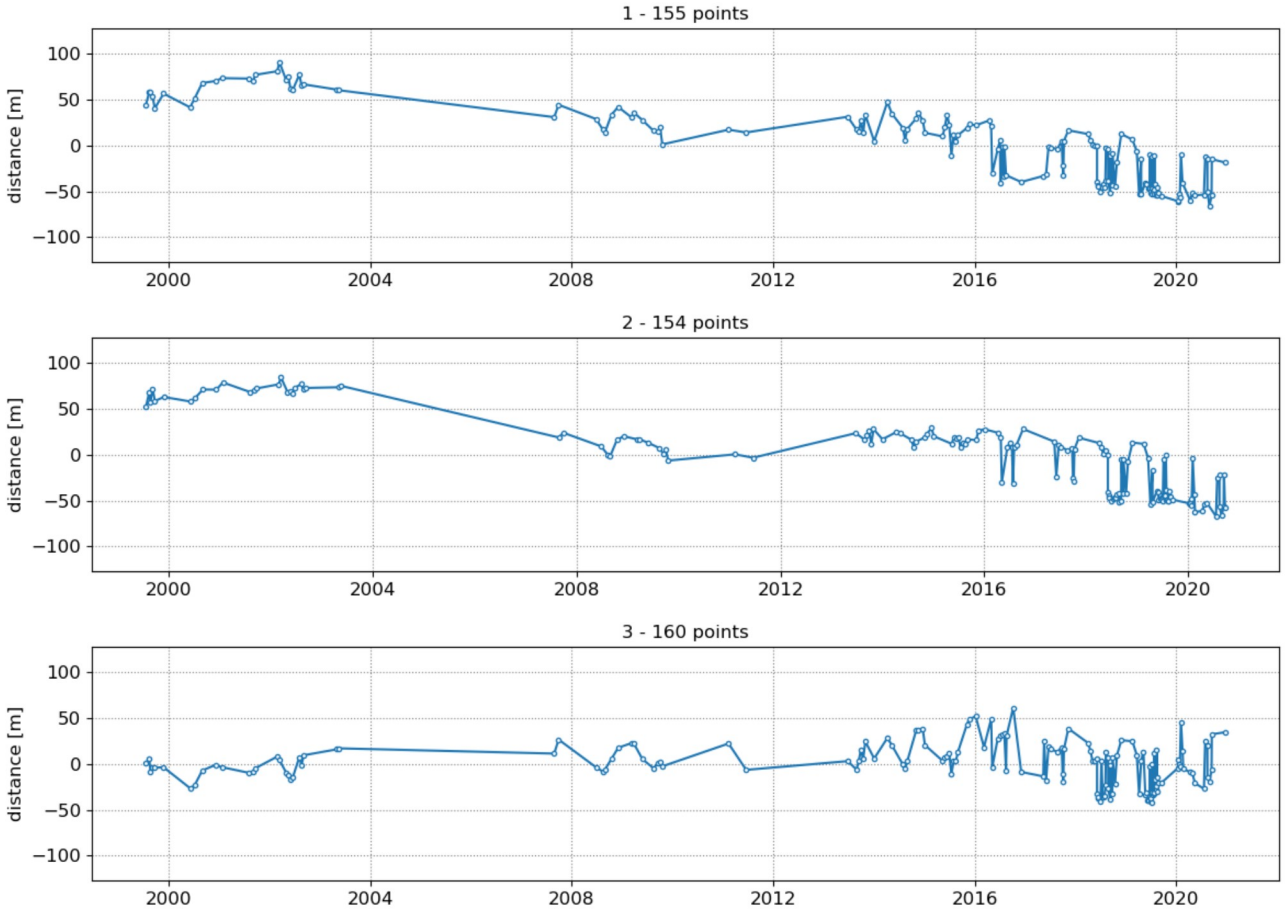
To obtain time-series of shoreline change we need to calculate the intersections between the 2D shorelines and the cross-shore transects, this can be done in the CoastSat toolbox but I provided here a more advanced method that deals with outliers and erroneous detections. As the accuracy

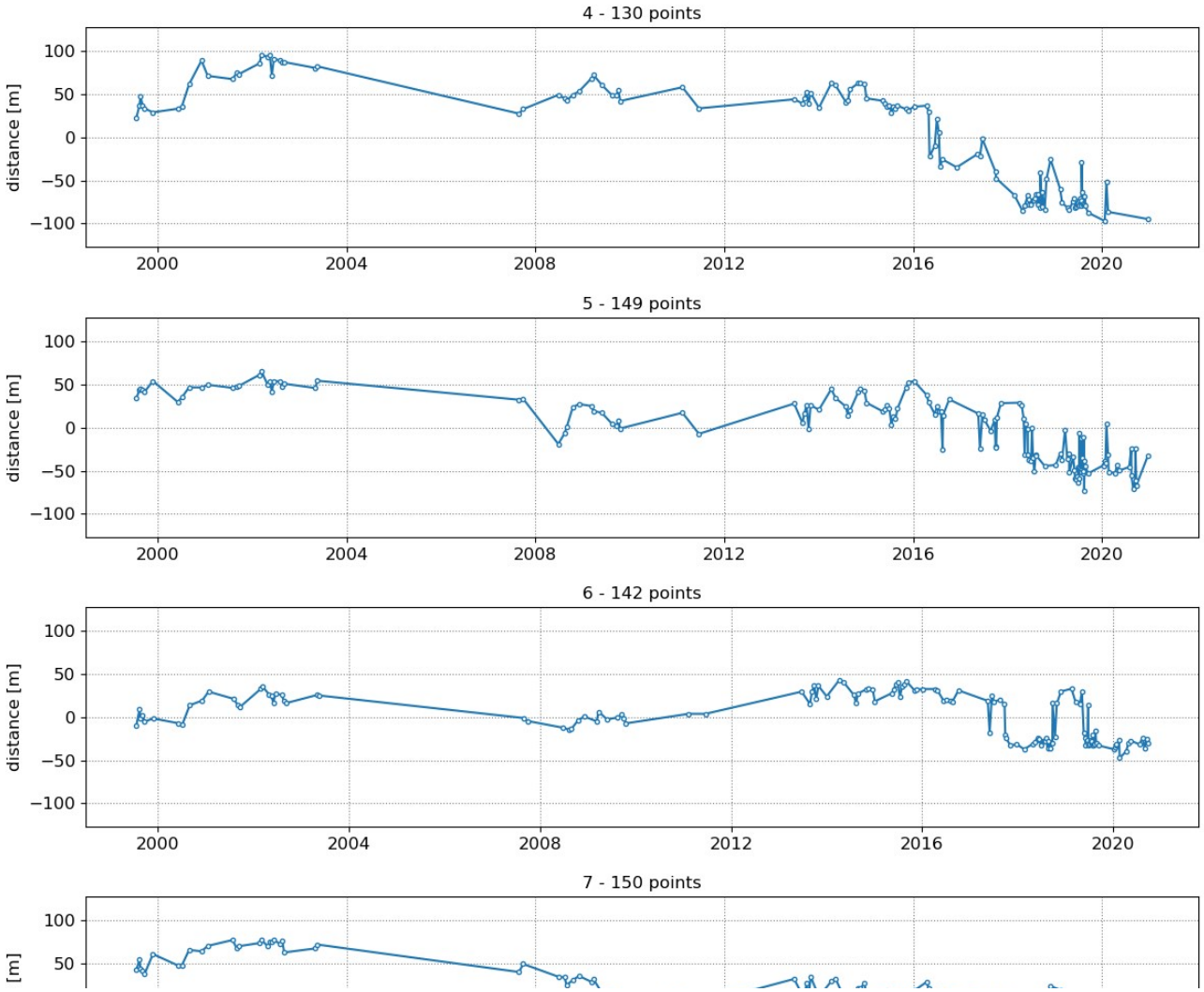
of the beach slope estimate will depend on the quality of the satellite-derived shorelines, it is important to get rid of large outliers as these will affect the slope estimates.

To remove outliers use the `max_cross_change` parameter to define the maximum cross-shore distance for despiking the time-series. Narrabeen-Collaroy is microtidal and storm-dominated, therefore the threshold was set at 40 m.

In [3]:
```
# a more robust method to compute intersections is provided here to avoid the presence of outliers in the time-series
settings_transects = { # parameters for shoreline intersections
                      'along_dist':          25,          # along-shore distance to use for intersection
                      'max_std':             15,          # max std for points around transect
                      'max_range':           30,          # max range for points around transect
                      'min_val':             -100,        # largest negative value along transect (landwards of transec
                      # parameters for outlier removal
                      'nan/max':             'auto',      # mode for removing outliers ('auto', 'nan', 'max')
                      'prc_std':             0.1,         # percentage to use in 'auto' mode to switch from 'nan' to 'm
                      'max_cross_change':    40,          # maximum cross-shore distance for despiking
                      }
# compute intersections [advanced version]
cross_distance = SDS_slope.compute_intersection(output, transects, settings_transects)
# remove outliers [advanced version]
cross_distance = SDS_slope.reject_outliers(cross_distance,output,settings_transects)
# plot time-series
SDS_slope.plot_cross_distance(output['dates'],cross_distance)

print(len(output['dates']))
```

```
1  - outliers removed 20
2  - outliers removed 23
3  - outliers removed 17
4  - outliers removed 26
5  - outliers removed 16
6  - outliers removed 34
7  - outliers removed 26
177
```





3. Get tide levels at the time of image acquisition

Now that we have the time-series of shoreline change, we need to obtain the tide level at the time of image acquisition for each data point. There are two options to get the tide levels:

- **Option 1:** Use a global tide model (FES2014 from AVISO) to get the modeled tide levels at the time of image acquisition
- **Option 2:** Provide your own file with measured/modeled tide levels

There are also some parameters to estimate the beach slope. You can change the trial beach slopes if the range does not correspond to the beach slope at your site by changing `slope_min` and `slope_max`. Do not change any of the other parameters.

In the section below the time-series of shoreline change are cropped between 1999 and 2000 as this is the period when 2 Landsat satellites are concurrently in orbit (providing a minimum sampling period of 8 days).

In [4]:
```
# slope estimation settings
days_in_year = 365.2425
seconds_in_day = 24*3600
settings_slope = {'slope_min':        0.035,                  # minimum slope to trial
                  'slope_max':        0.2,                    # maximum slope to trial
                  'delta_slope':      0.005,                  # slope increment
                  'date_range':       [1999,2020],            # range of dates over which to perform the analysis
                  'n_days':           8,                      # sampling period [days]
                  'n0':               50,                     # parameter for Nyquist criterium in Lomb-Scargle trans
                  'freqs_cutoff':     1./(seconds_in_day*30), # 1 month frequency
                  'delta_f':          100*1e-10,              # deltaf for identifying peak tidal frequency band
                  'prc_conf':         0.05,                   # percentage above minimum to define confidence bands i
                  }
settings_slope['date_range'] = [pytz.utc.localize(datetime(settings_slope['date_range'][0],5,1)),
                                pytz.utc.localize(datetime(settings_slope['date_range'][1],1,1))]
beach_slopes = SDS_slope.range_slopes(settings_slope['slope_min'], settings_slope['slope_max'], settings_slope['delta

# clip the dates between 1999 and 2020 as we need at least 2 Landsat satellites
idx_dates = [np.logical_and(_>settings_slope['date_range'][0],_<settings_slope['date_range'][1]) for _ in output['dat
dates_sat = [output['dates'][_] for _ in np.where(idx_dates)[0]]
for key in cross_distance.keys():
    cross_distance[key] = cross_distance[key][idx_dates]

print(len(dates_sat))
```

```
158
```

Option 1: get tide levels from FES2014

You will need to install FES2014 following the instructions provided here. Information about this global tide model can be found on AVISO's website.

In the section below the tide level corresponding to each date in `dates_sat` is computed from the model in a numpy.array named `tide_sat`.

In [60]:
```
# Option 1. if FES2014 global tide model is setup
import pyfes
# point to the folder where you downloaded the .nc files
filepath = r'C:\Users\z5030440\OneDrive - UNSW\fes-2.9.1-Source\data\fes2014'
config_ocean = os.path.join(filepath, 'ocean_tide.ini') # change to ocean_tide.ini
config_load =  os.path.join(filepath, 'load_tide.ini')  # change to load_tide.ini
ocean_tide = pyfes.Handler("ocean", "io", config_ocean)
load_tide = pyfes.Handler("radial", "io", config_load)

# coordinates of the location (always select a point 1-2km offshore from the beach)
# if the model returns NaNs, change the location of your point further offshore.
coords = [151.332209, -33.723772]
# get tide time-series with 15 minutes intervals
time_step = 15*60
dates_fes, tide_fes = SDS_slope.compute_tide(coords,settings_slope['date_range'],time_step,ocean_tide,load_tide)
# get tide level at time of image acquisition
tide_sat = SDS_slope.compute_tide_dates(coords, dates_sat, ocean_tide, load_tide)

# plot tide time-series
fig, ax = plt.subplots(1,1,figsize=(12,3), tight_layout=True)
ax.set_title('Sub-sampled tide levels')
ax.grid(which='major', linestyle=':', color='0.5')
ax.plot(dates_fes, tide_fes, '-', color='0.6')
ax.plot(dates_sat, tide_sat, '-o', color='k', ms=4, mfc='w',lw=1)
ax.set_ylabel('tide level [m]')
ax.set_ylim(SDS_slope.get_min_max(tide_fes));
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_17120/3912639784.py in <module>
      1 # Option 1. if FES2014 global tide model is setup
----> 2 import pyfes
      3 # point to the folder where you downloaded the .nc files
      4 filepath = r'C:\Users\z5030440\OneDrive - UNSW\fes-2.9.1-Source\data\fes2014'
      5 config_ocean = os.path.join(filepath, 'ocean_tide.ini') # change to ocean_tide.ini

ModuleNotFoundError: No module named 'pyfes'
```

Option 2: load the tide levels from your own file

If you prefer to use measured water levels or astronomical tides from your own model, you can provide your own file with the tide levels associated with the dates at which the shorelines where mapped (`dates_sat`). An example is provided below, you will need to create a numpy.array called `tides_sat` which contains an array of tide levels corresponding to each date in `dates_sat`.

In [6]:

```
# Option 2. load tide levels corresponding to "dates_sat" from a file
#with open(os.path.join('example_data', 'NARRA' + '_tide' + '.pkl'), 'rb') as f:
#    tide_data = pickle.load(f)
#tides_sat = tide_data['tide']
#print(tides_sat)
#print(len(tides_sat))
#print(tide_data)
#print(len(dates_sat))

import pandas as pd
# altering this because I'm importing a CSV
# load the measured tide data
#filepath = os.path.join(os.getcwd(),'data','Pekalongan_longer_time','overall_tide_data_output.csv')
dates_fes = pd.read_csv(os.path.join(os.getcwd(),'data','Pekalongan_longer_time','date_fes_ts.csv'))
tide_fes = pd.read_csv(os.path.join(os.getcwd(),'data','Pekalongan_longer_time','tide_fes.csv'))
tide_sat = pd.read_csv(os.path.join(os.getcwd(),'data','Pekalongan_longer_time','tide_sat_with_sentinel.csv'))
#dates_sat = pd.read_csv(os.path.join(os.getcwd(),'data','Pekalongan_longer_time','date_sat_ts_with_sentinel.csv'))

#dates_ts = [pytz.utc.localize(_) for _ in dates_ts]
#tides_sat = np.array(tide_data['tides_sat'])

#print(dates_sat)
#print('dates_sat type:')
#print(type(dates_sat))
#print('')
print(tide_sat)
```

```
     1.866252730540491137e-02
0                    0.323416
1                    0.054625
2                    0.050293
3                    0.089941
4                    0.032719
5                    0.143697
6                    0.130282
7                    0.357513
8                   -0.239897
9                   -0.063979
10                   0.208984
11                   0.238275
12                   0.342995
13                  -0.024996
14                  -0.105382
15                  -0.065435
16                  -0.029856
17                  -0.052480
18                  -0.014498
19                   0.121833
20                   0.236508
21                   0.237951
22                  -0.000955
23                  -0.000834
24                   0.228652
25                   0.221642
26                   0.096151
27                   0.142788
28                   0.226806
29                   0.068845
..                        ...
127                  0.013965
128                 -0.042086
129                 -0.026091
130                 -0.008565
131                  0.017340
132                  0.064877
133                  0.059089
134                  0.136848
135                  0.153340
136                  0.137369
137                  0.006498
138                  0.020688
139                  0.134041
140                  0.256662
141                  0.012003
142                  0.004546
143                  0.209044
144                  0.195472
145                  0.054556
146                  0.073155
147                  0.188955
148                  0.297315
149                 -0.000570
150                  0.154268
```

```
151                  0.219892
152                  0.165958
153                  0.004378
154                  0.174091
155                  0.054286
156                 -0.316879

[157 rows x 1 columns]
```

4. Peak tidal frequency

Find the peak tidal frequency, frequency band at which the energy is the largest in the subsampled tide level time-series.

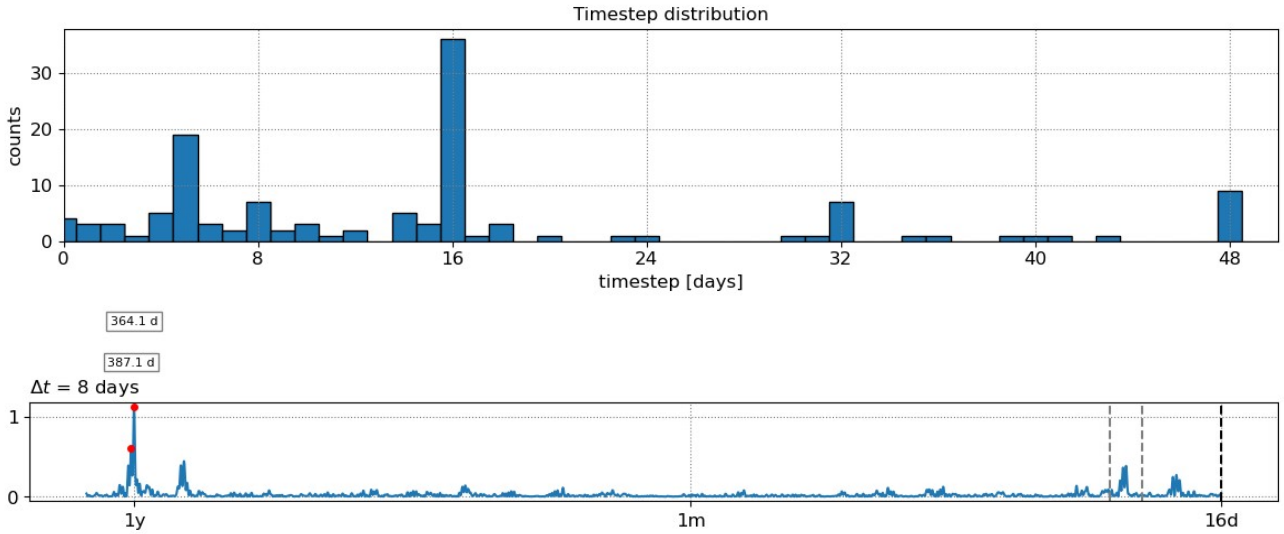
Most sites will have a minimum sampling period of 8 days, but it can happen that because of overlapping images at some sites, a minimum sampling period of 7 days is achieved, then you can use 7 days instead of 8 by setting `settings_slope['n_days] = 7` . Don't use a sampling period of less than 7 days. If the plot of timestep distribution doesn't show a peak at 7 or 8 days, you will not be able to apply this technique as you don't have enough images.

In [7]:

```
# plot time-step distribution
t = np.array([_.timestamp() for _ in dates_sat]).astype('float64')
#t = dates_sat
#t = pd.DataFrame.to_numpy(t)
#print(t)
#print(type(t))
#print(t.size)
#print(t.ndim)

#print('t reformatted:')
#t = np.squeeze(t, axis=1)
#manually add back in the first element - MAKE SURE TO DO THIS AGAIN IF REVISING - LOOK HERE!!!!!!!!!!
#t = np.append([932437551],t)
#print(t)
#print(type(t))
#print(t.size)
#print(t.ndim)

delta_t = np.diff(t,axis=0)
#delta_t = pd.DataFrame.diff(t)
fig, ax = plt.subplots(1,1,figsize=(12,3), tight_layout=True)
ax.grid(which='major', linestyle=':', color='0.5')

bins = np.arange(np.min(delta_t)/seconds_in_day, np.max(delta_t)/seconds_in_day+1,1)-0.5
ax.hist(delta_t/seconds_in_day, bins=bins, ec='k', width=1);
ax.set(xlabel='timestep [days]', ylabel='counts',
       xticks=settings_slope['n_days']*np.arange(0,20),
       xlim=[0,50], title='Timestep distribution');

# find tidal peak frequency
settings_slope['n_days'] = 8

#print('debug')

#print('dates:')
#print(type(dates_sat))
#print('tides:')
#print(tide_sat)
#print('settings:')
#print(type(settings_slope))

#convert tides_sat to numpy array
tide_sat = pd.DataFrame.to_numpy(tide_sat)

#print('debug II')

#print('dates:')
#print(type(dates_sat))
#print('tides reformatted:')
tide_sat = np.squeeze(tide_sat, axis=1)
#manually add back in the first element - MAKE SURE TO DO THIS AGAIN IF REVISING - LOOK HERE!!!!!!!!!!
tide_sat = np.append([0.0186625273054049],tide_sat)
#print(tide_sat)
#print(type(tide_sat))
#print('tide_sat.ndim:')
#print(tide_sat.ndim)
#print('checking sizes:')
#print('tide_sat.size:')
#print(tide_sat.size)
#print('dates_sat.size:')
#print(len(dates_sat))
#print('settings:')
#print(type(settings_slope))

#settings_slope['freqs_max'] = SDS_slope.find_tide_peak(dates_sat,tide_sat,settings_slope)
#attempting to feed t in here instead, as it looks like the same thing is done to date_sat in the function
# as it is in the script
print(tide_sat)
settings_slope['freqs_max'] = SDS_slope.find_tide_peak(t,tide_sat,settings_slope)
```

```
[ 0.01866253  0.32341612  0.05462514  0.05029257  0.08994052  0.03271876
  0.14369706  0.13028194  0.35751328 -0.23989705 -0.0639791   0.20898382
  0.23827481  0.34299534 -0.02499555 -0.10538244 -0.06543527 -0.02985552
 -0.05247962 -0.0144985   0.12183342  0.23650801  0.23795073 -0.00095465
 -0.00083447  0.22865168  0.22164214  0.09615136  0.14278782  0.2268059
  0.06884548 -0.27590042 -0.06748204 -0.07285752  0.13713191  0.0093569
  0.14684363  0.07631463  0.08202384 -0.22018981  0.21070605  0.29684962
 -0.04030803 -0.07859099 -0.14962087 -0.22768435 -0.15854083 -0.00592403
 -0.13229441 -0.01415223  0.22963288  0.28168572  0.08459388 -0.27207069
 -0.16793005 -0.13779396 -0.07549426 -0.08179683  0.03746626  0.12372994
  0.15376601  0.25404841  0.30043379  0.2114873   0.23696476 -0.22857022
 -0.29606824 -0.1197353  -0.04662072 -0.15341999 -0.09329417  0.13568147
  0.08151467  0.12954821  0.28503347  0.17568958  0.09579585  0.29473209
```

```
 0.09017496  0.26863103 -0.07679033 -0.19126893  0.05923882 -0.06949744
-0.0374455   0.00586736  0.12045701  0.32159454  0.18617046  0.13296954
-0.14775471  0.02911336  0.06885685 -0.09454887 -0.30636122 -0.24312375
-0.18889168 -0.07188908 -0.00931377 -0.02617663 -0.04485929 -0.0313826
-0.05867198 -0.0383979  -0.02593801  0.05963662  0.21360472  0.16700144
 0.04875668  0.3073235   0.17660581  0.21238524  0.04883367  0.11993743
-0.02449851  0.19529041  0.17192988  0.21158942  0.32163507  0.11468713
 0.15034632 -0.11674302 -0.04084555 -0.31998033 -0.13849592 -0.23728638
-0.11207567 -0.29927974  0.01396474 -0.04208625 -0.0260911  -0.00856459
 0.01734021  0.06487699  0.05908938  0.13684812  0.1533399   0.13736947
 0.00649774  0.02068833  0.13404071  0.25666196  0.01200261  0.00454554
 0.20904353  0.19547166  0.05455639  0.07315517  0.18895513  0.29731512
-0.00057045  0.15426807  0.21989245  0.16595819  0.00437812  0.17409071
 0.05428611 -0.31687916]
```



5. Estimate the beach slope

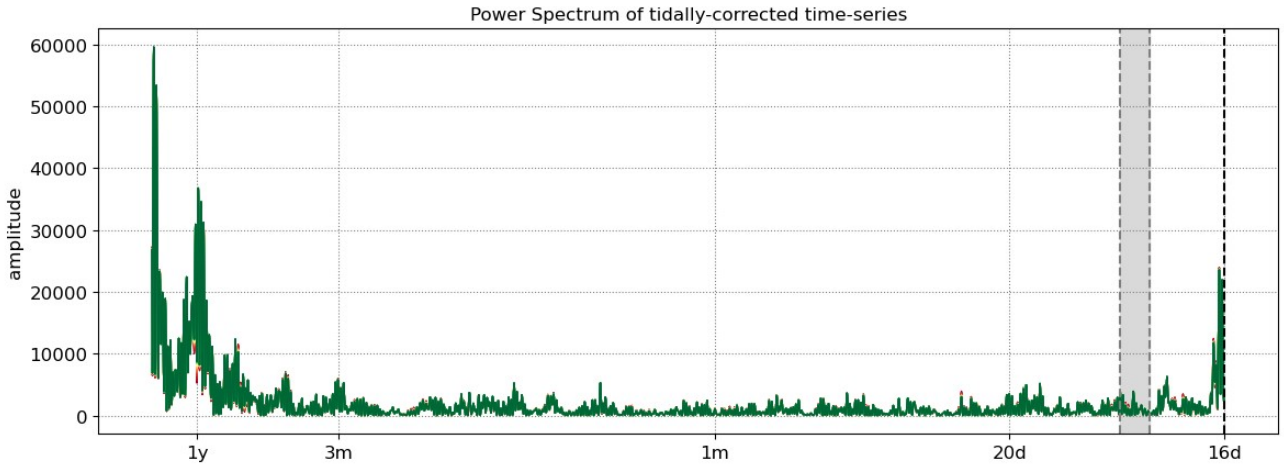
The beach slope along each transect is estimated by finding the slope that, when used for tidal correction, minimises the energy in the peak tidal frequency band. Based on our validation study, this slopes corresponds to the beach-face slope between mean sea level (MSL) and mean high water springs (MHWS).

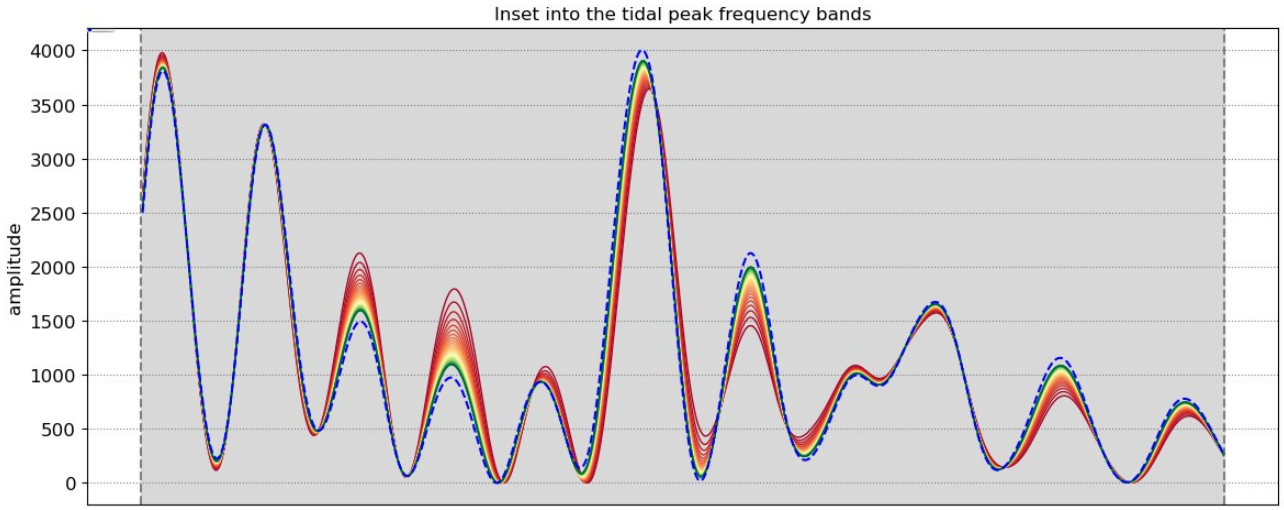
In [13]:

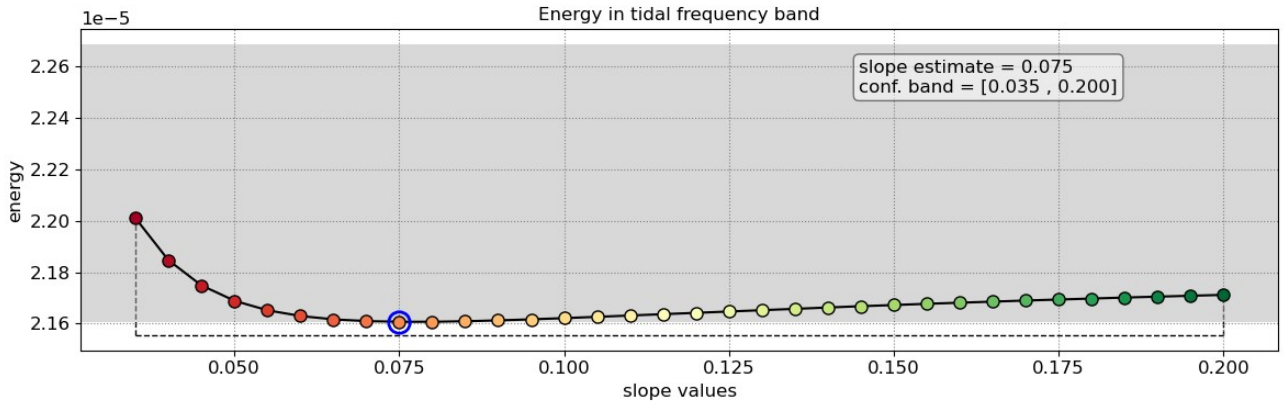
```
#import sys
#import traceback

#class TracePrints(object):
#  def __init__(self):
#    self.stdout = sys.stdout
#  def write(self, s):
#    self.stdout.write("Writing %r\n" % s)
#    traceback.print_stack(file=self.stdout)

#sys.stdout = TracePrints()
print('composite before')
print(len(composite))

#estimate beach-face slopes along the transects
slope_est, cis = dict([]), dict([])
for key in cross_distance.keys():
    # remove NaNs
    idx_nan = np.isnan(cross_distance[key])
    dates = [dates_sat[_] for _ in np.where(~idx_nan)[0]]
    tide = tide_sat[~idx_nan]
    composite = cross_distance[key][~idx_nan]

    #composite = cross_distance[key]

#dates = dates_sat
#tide = tide_sat

print(idx_nan)
print(len(tide))
print(len(dates))
print('composite after')
print(len(composite))


#print(cross_distance)
#print(cross_distance["1"])
#print(composite)

for key in cross_distance.keys():
    # apply tidal correction
    tsall = SDS_slope.tide_correct(composite,tide,beach_slopes)
    title = 'Transect %s'%key
    SDS_slope.plot_spectrum_all(dates,composite,tsall,settings_slope, title)
    slope_est[key],cis[key] = SDS_slope.integrate_power_spectrum(dates,tsall,settings_slope)


    print('Beach slope at transect %s: %.3f'%(key, slope_est[key]))
```

```
composite before
158
[False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False  True False False False
 False False  True False  True False  True False False  True  True False
  True False False False  True False False False False  True False  True
  True False False  True False  True False False False False False  True
  True False False False False False  True False  True  True False False
  True False False False  True  True False  True False False False False
  True False False False False False False False False False False False
 False False False False False False False  True False False False False
  True  True]
132
132
composite after
132
Beach slope at transect 1: 0.075
Beach slope at transect 2: 0.075
Beach slope at transect 3: 0.075
Beach slope at transect 4: 0.075
Beach slope at transect 5: 0.075
Beach slope at transect 6: 0.075
Beach slope at transect 7: 0.075
```

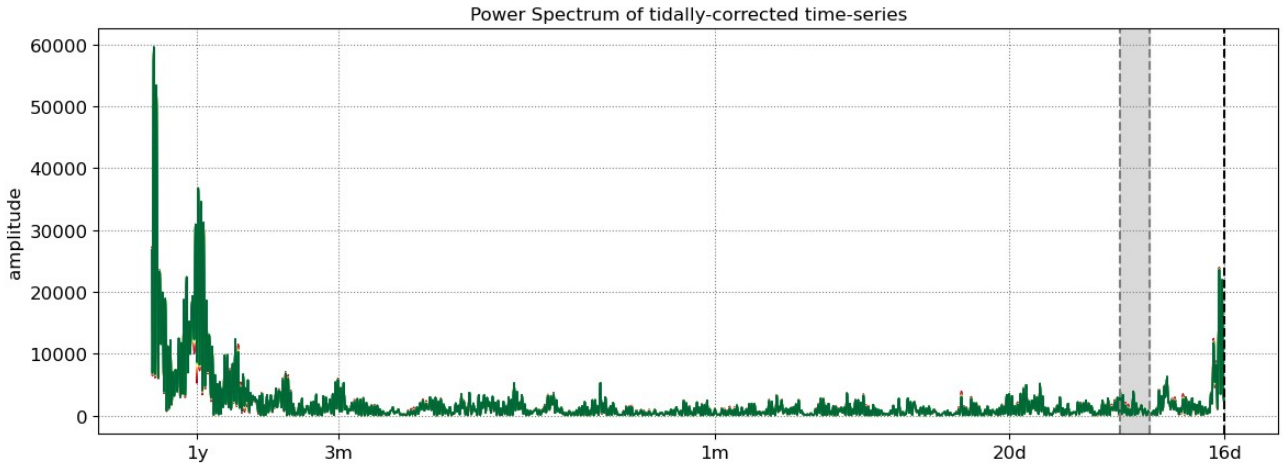
Transect 1

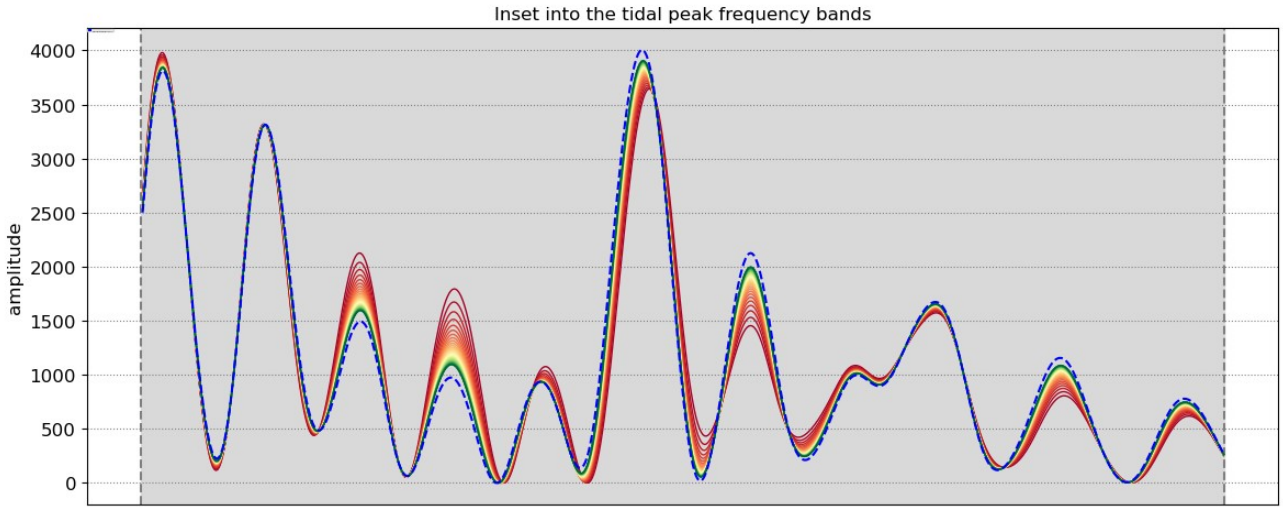

Power Spectrum of tidally-corrected time-series
amplitude
60000
50000
40000
30000
20000
10000
0
1y
3m
1m
20d
16d

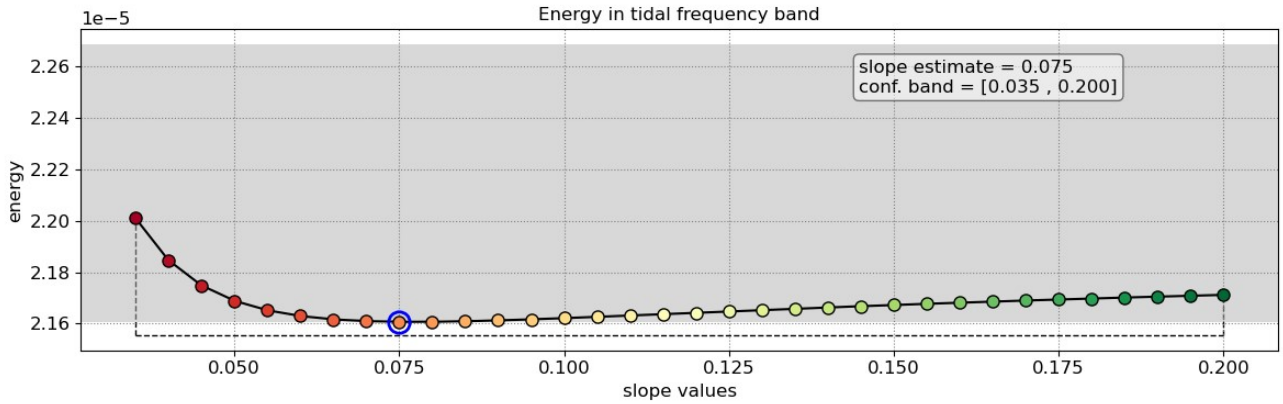

Inset into the tidal peak frequency bands
amplitude
4000
3500
3000
2500
2000
1500
1000
500
0


Energy in tidal frequency band
1e−5
slope estimate = 0.075
conf. band = [0.035 , 0.200]
energy
2.26
2.24
2.22
2.20
2.18
2.16
0.050
0.075
0.100
0.125
0.150
0.175
0.200
slope values

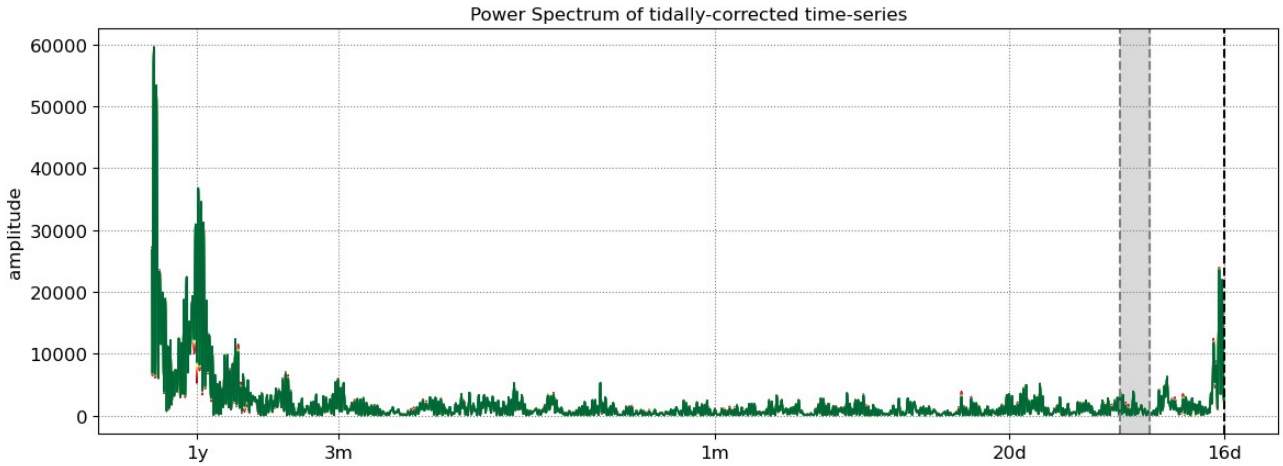
Transect 2

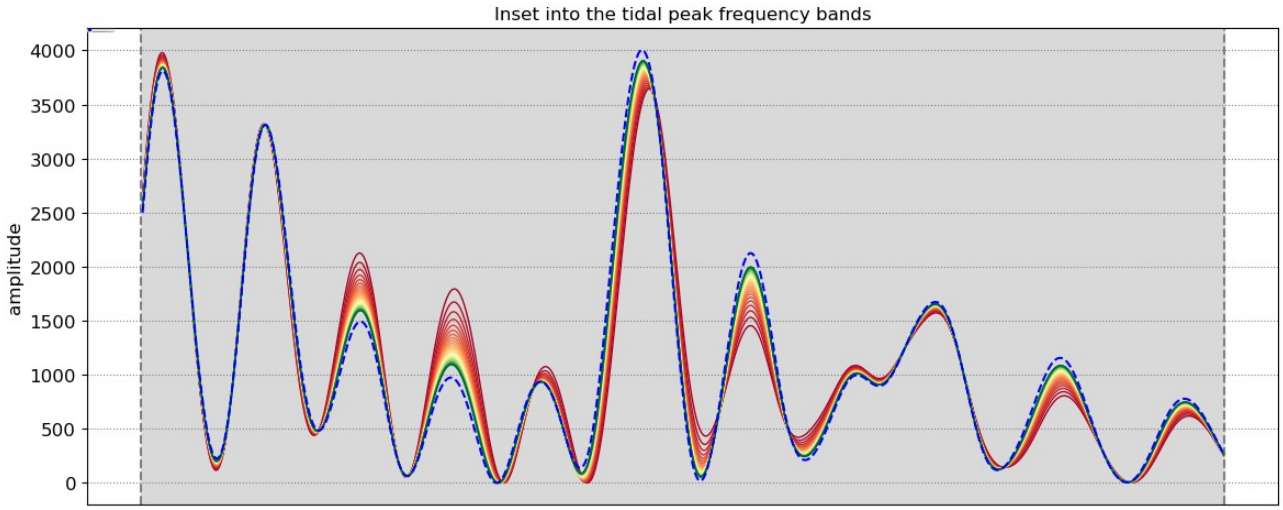

Power Spectrum of tidally-corrected time-series
amplitude
60000
50000
40000
30000
20000
10000
0
1y
3m
1m
20d
16d

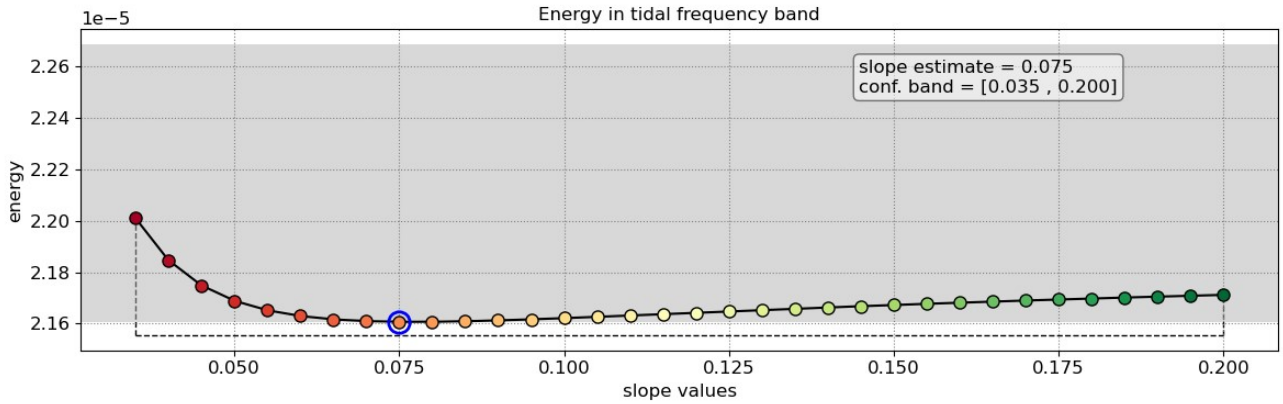

Inset into the tidal peak frequency bands
amplitude
4000
3500
3000
2500
2000
1500
1000
500
0


Energy in tidal frequency band
1e−5
slope estimate = 0.075
conf. band = [0.035 , 0.200]
energy
2.26
2.24
2.22
2.20
2.18
2.16
0.050
0.075
0.100
0.125
0.150
0.175
0.200
slope values

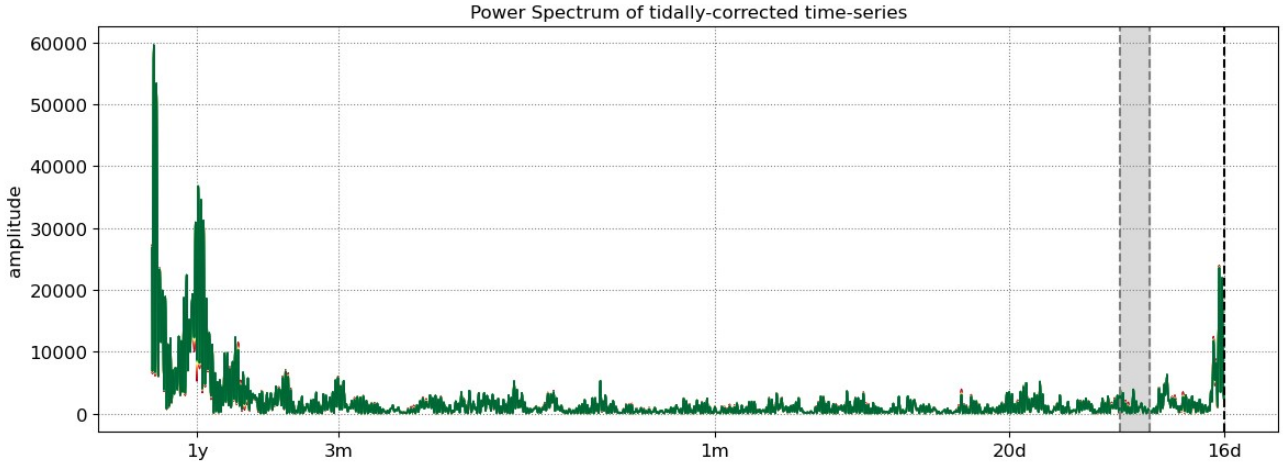
Transect 3

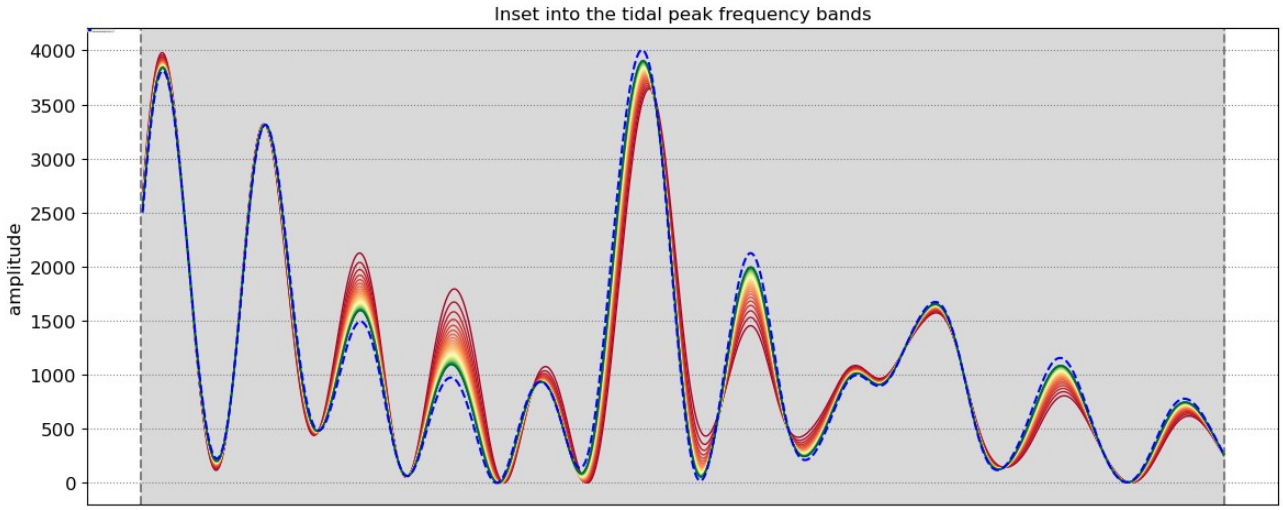

Power Spectrum of tidally-corrected time-series
amplitude
60000
50000
40000
30000
20000
10000
0
1y
3m
1m
20d
16d

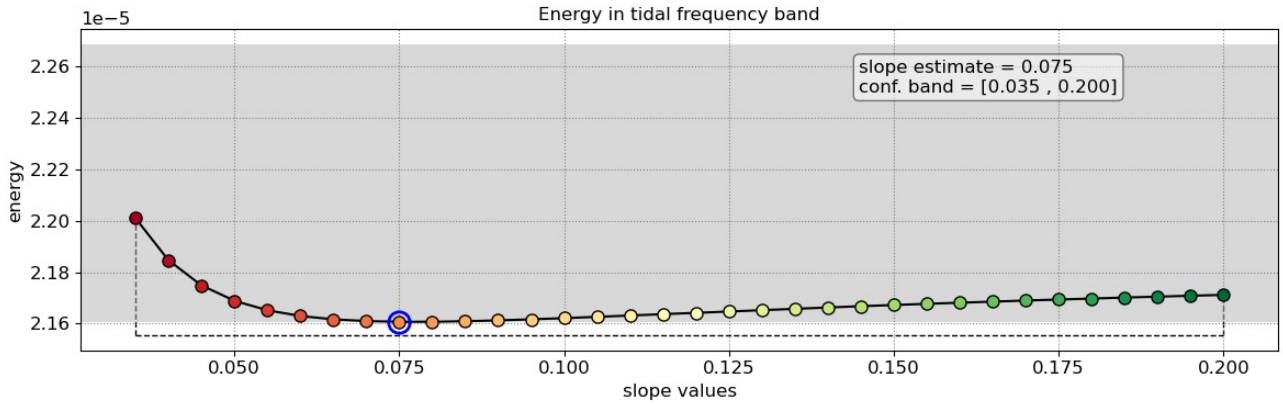

Inset into the tidal peak frequency bands
amplitude
4000
3500
3000
2500
2000
1500
1000
500
0


Energy in tidal frequency band
1e−5
slope estimate = 0.075
conf. band = [0.035 , 0.200]
energy
2.26
2.24
2.22
2.20
2.18
2.16
0.050
0.075
0.100
0.125
0.150
0.175
0.200
slope values

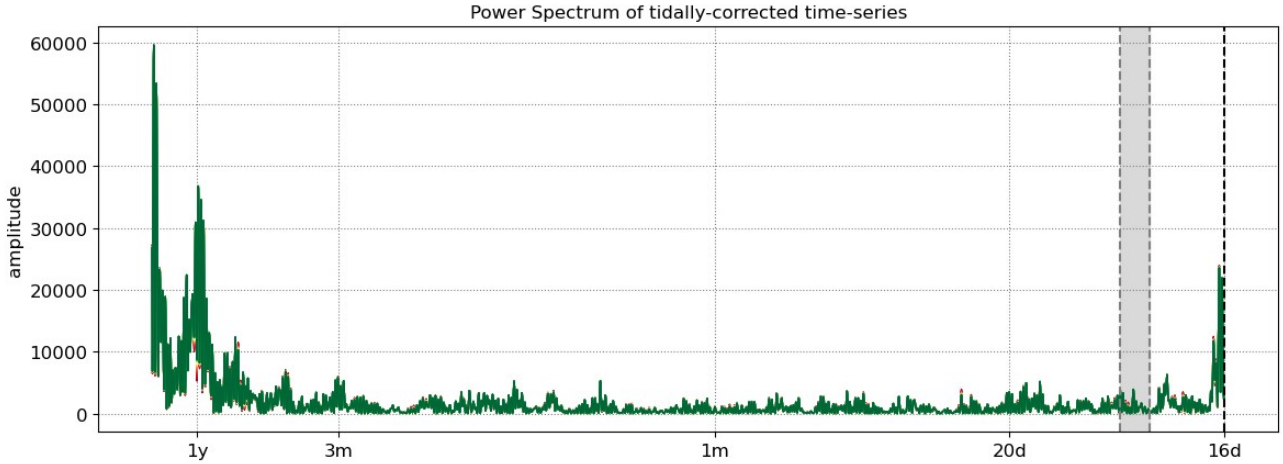
Transect 4

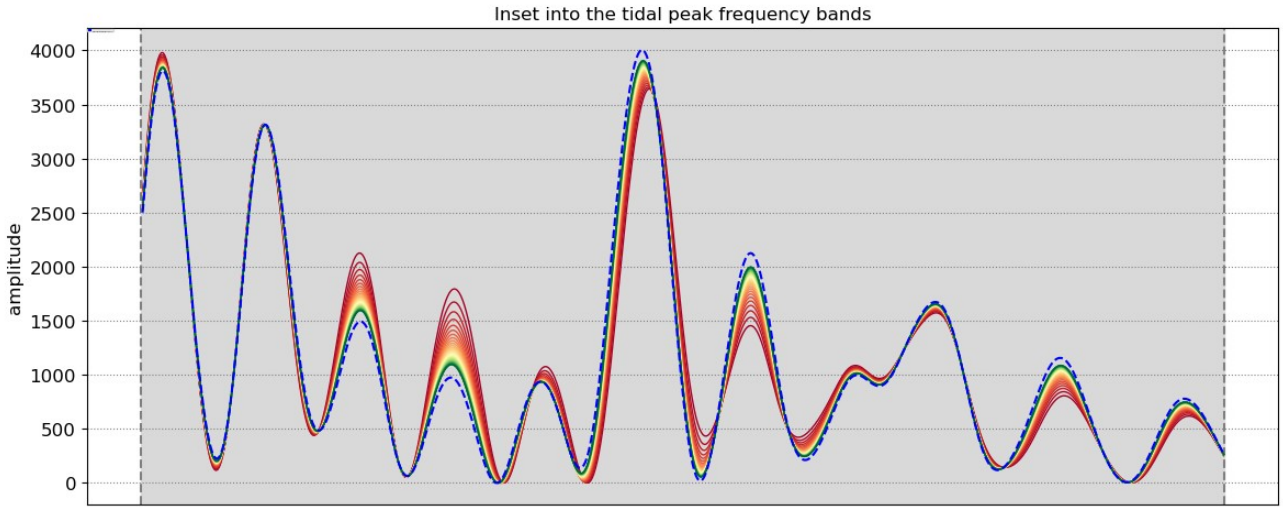

Power Spectrum of tidally-corrected time-series
amplitude
60000
50000
40000
30000
20000
10000
0
1y
3m
1m
20d
16d

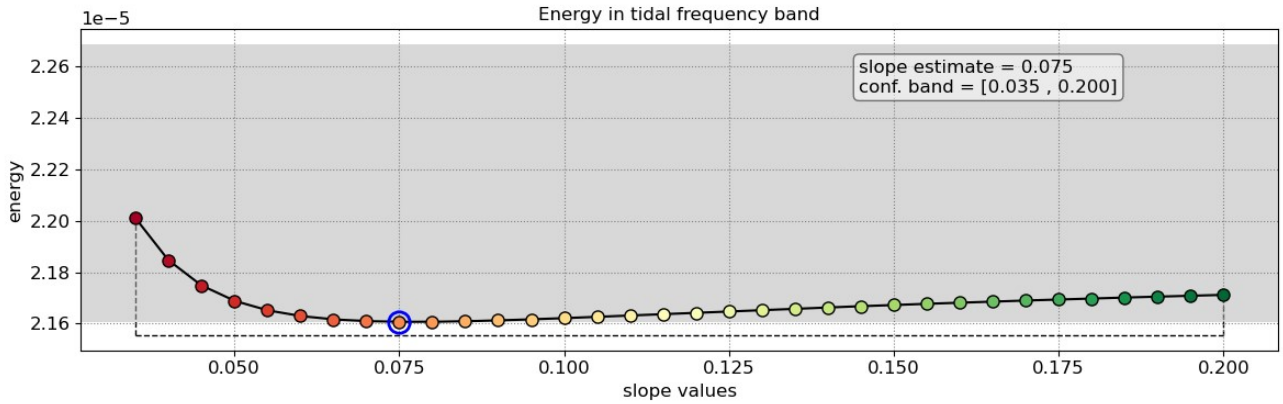

Inset into the tidal peak frequency bands
amplitude
4000
3500
3000
2500
2000
1500
1000
500
0


Energy in tidal frequency band
1e−5
slope estimate = 0.075
conf. band = [0.035 , 0.200]
energy
2.26
2.24
2.22
2.20
2.18
2.16
0.050
0.075
0.100
0.125
0.150
0.175
0.200
slope values

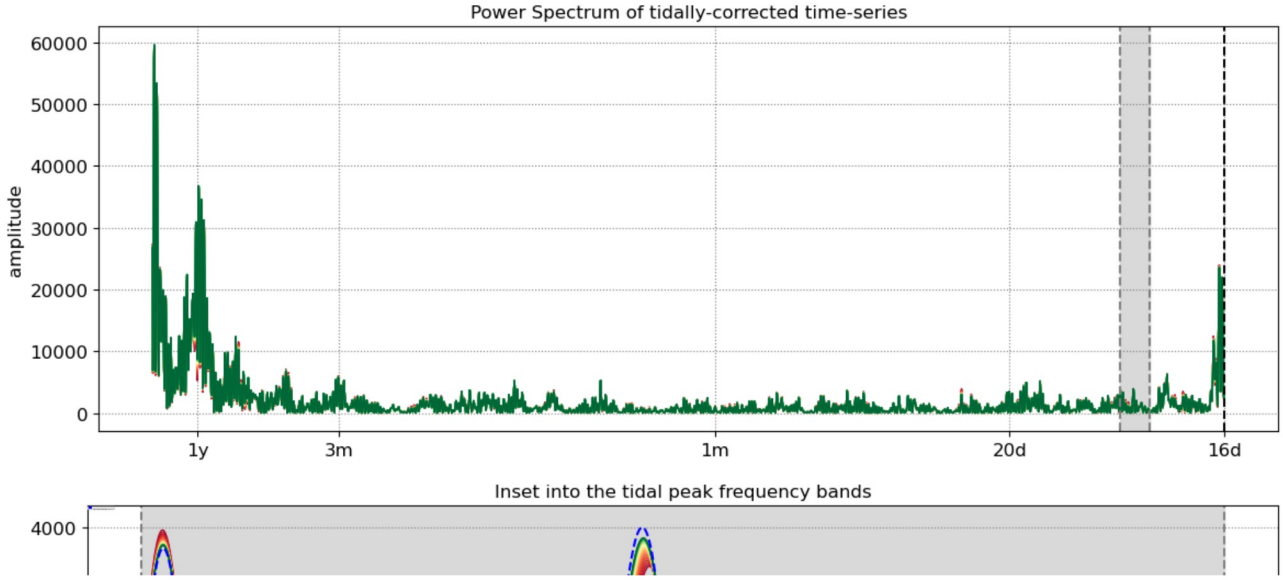
Transect 5


Power Spectrum of tidally-corrected time-series
amplitude
60000
50000
40000
30000
20000
10000
0
1y
3m
1m
20d
16d


Inset into the tidal peak frequency bands
amplitude
4000
3500
3000
2500
2000
1500
1000
500
0


Energy in tidal frequency band
1e−5
slope estimate = 0.075
conf. band = [0.035 , 0.200]
energy
2.26
2.24
2.22
2.20
2.18
2.16
0.050
0.075
0.100
0.125
0.150
0.175
0.200
slope values

Transect 6



In [ ]:

In [ ]: