```
# ----------------------#
# Files and directories
# ----------------------#
## -- Get the user home directory
from os.path import expanduser
import os
home = expanduser("/u/GoM_4dvar/SWOT-cal-val/model/OoT/raw/")
# ------ Directory that contains orbit file:
dir_setup = os.path.join(home, 'SWOT_OoT', 'swotsimulator', 'data')
# ------ Directory that contains your own inputs:
indatadir = os.path.join(home, 'ssha')
# ------ Directory that contains your outputs:
outdatadir = os.path.join(home, 'SWOT_OoT', 'swot_output')
# ------ Orbit file:
# Order of columns (lon, lat, time) in the orbit file
# (default is (1, 2, 0) with order_orbit_col = None)
order_orbit_col = None
# Name of the orbit file
satname = "science"
filesat = os.path.join(dir_setup, 'ephem_science_sept2015_ell.txt')
# ------ Number of days in one cycle
satcycle = 20.86455
# ------ Satellite elevation
sat_elev = 891 * 10**3
# ------ Name of the configuration (to build output files names)
config = "OoT"
#Number of processors to be used
proc_number = 1
# ------ Deactivate printing of progress bar to avoid huge log
progress_bar = False

# ----------------------#
# SWOT swath parameters
# ----------------------#
# ------ Satellite grid file root name:
#      (Final file name is root_name_[numberofpass].nc)
filesgrid = os.path.join(outdatadir, '{}_{}_grid'.format(config,satname))
# ------ Force the computation of the satellite grid:
makesgrid = False
# ------ Give a subdomain if only part of the model is needed:
#      (modelbox=[lon_min, lon_max, lat_min, lat_max])
#      (If modelbox is None, the whole domain of the model is considered)
modelbox = None  # [230.144,234.598,42.27,47.8283]
# ------ Distance between the nadir and the end of the swath (in km):
halfswath = 60.
# ------ Distance between the nadir and the beginning of the swath (in
km):
halfgap = 10.
# ------ Along track resolution (in km):
delta_al = 2.
# ------ Across track resolution (in km):
delta_ac = 2.
# ------ Shift longitude of the orbit file if no pass is in the domain
#        (in degree): Default value is None (no shift)
```

```
shift_lon = None
# ------ Shift time of the satellite pass (in day):
#        Default value is None (no shift)
shift_time = None

# ----------------------#
# Model input parameters
# ----------------------#
# ------ List of model files:
#      (The first file contains the grid and is not considered as model
data)
#        To generate the noise alone, file_input = None
#        and specify region in modelbox
file_input = os.path.join(indatadir, 'list_of_file.txt')
# ------ Type of model data:
#      (Optional, default is NETCDF_MODEL and reads netcdf3 and netcdf4
files)
#      (Other options are ROMS, NEMO and CLS to read Nemo, roms or CLS)
model = 'NETCDF_MODEL'
# ------ First time of the model
first_time = '2019-01-01T01:00:00Z'
# ------ Grid file name
file_grid_model = os.path.join(indatadir,
'ncom_1_2019010100_00010000.nc')
# ------ Type of grid:
#        'regular' or 'irregular', if 'regular' only 1d coordinates
#        are extracted from model
grid = 'irregular'
# ------ Specify SSH variable:
var = 'ssha'
# ------ Specify list of variables, using the format: {key:
[variable_name,
#        file_suffix], ...}, should contain at least the key 'ssh_true':
list_input_var = None
# ------ Specify factor to convert SSH values in m:
SSH_factor = 1.
# ------ Specify longitude variable:
lon = 'lon'
# ------ Specify latitude variable:
lat = 'lat'
# ------ Specify number of time in each file:
dim_time = 1
# ------ Time step between two model outputs (in days):
timestep = 0.04166667
# ------ Number of outputs to consider:
#        (timestep*nstep=total number of days)
nstep = 17519
# ------ Not a number value:
model_nan = 'NaN'

# ----------------------#
# SWOT output files
# ----------------------#
# ------ Output file root name:
```

```
#      (Final file name is root_name_c[cycle]_p[pass].nc
file_output = os.path.join(outdatadir, '{}_{}'.format(config, satname))
# ------ Interpolation of the SSH from the model (if grid is irregular
and
#         pyresample is not installed:
#        (either 'linear' or 'nearest', use 'nearest' for large region
#        as it is faster and use less memory.)
interpolation = 'nearest'
# ------ Save variables with mockup variables ('mockup') swotsimulator
#        variables ('classic', default behaviour) or in expert mode
('expert')
save_variables = 'classic'

# ----------------------#
# SWOT error parameters
# ----------------------#
# ------ File containing random coefficients to compute and save
#      random error coefficients so that runs are reproducible:
#        If file_coeff is specified and does not exist, file is created
#      If you don't want runs to be reproducible, file_coeff is set to
None
file_coeff = None
# file_coeff = os.path.join(outdatadir,'Random_coeff.nc')
# ------ KaRIN noise (True to compute it):
karin = True
# ------ KaRIN file containing spectrum for several SWH:
karin_file = os.path.join(dir_setup, 'karin_noise.nc')
# ------ SWH for the region:
#        if swh greater than 7 m, swh is set to 7
swh = 2.0
# ------ Number of km of random coefficients for KaRIN noise (recommended
nrandkarin=1000):
nrandkarin = 1000

## -- Other instrument error (roll, phase, baseline dilation, timing)
## ---------------------------------------------------------------
# -- Compute nadir (True or False):
nadir = True
# ------ File containing spectrum of instrument error:
file_inst_error = os.path.join(dir_setup,
"global_sim_instrument_error.nc")
# ------ Number of random realisations for instrumental and geophysical
error
#        (recommended ncomp=2000), ncomp1d is used for 1D spectrum, and
ncomp2d
#        is used for 2D spectrum (wet troposphere computation):
ncomp1d = 4000
ncomp2d = 2000
# ------ Cut off frequency:
#      (Use lambda_cut=40000km for cross-calibration)
lambda_cut = 20000
lambda_max = lambda_cut
# ------ Save entire rando signal instead of random coefficients. Enables
a better randomness
```

```
savesignal = True
# ------ If savesignal is True, enter number of pseudo-period of
superimposed
#        signals and repeat length
npseudoper = 30. # Number of pseudo period of superimposed signals.
len_repeat = 40000*14*50.
# ------ Roll error (True to compute it):
roll = True
# ------ Phase error (True to compute it):
phase = True
# ------ Baseline dilation error (True to compute it):
baseline_dilation = True
# ------ Timing error (True to compute it):
timing = True

## -- Geophysical error
## ----------------------
# ------ Wet tropo error (True to compute it):
wet_tropo = True
# ------ Beam print size (in km):
#        Gaussian footprint of sigma km
sigma = 8.
# ------ Number of beam used to correct wet_tropo signal (1, 2 or
'both'):
nbeam = 2
# ------ Beam position if there are 2 beams (in km from nadir):
beam_pos_l = -35.
beam_pos_r = 35.
# ------ Sea State Bias error (Not implemented yet):
ssb = False
# (ssb not implemented yet)
```