Ocean simulator for science applications

Lucile Gaultier, Clément Ubelmann, Frederic Briol, Robin Chevrier
With support from Jinbo Wang

SWOT simulator for Ocean Science

- General overview
- Generation of parametrized errors : baseline (current version) and Best Estimates (option available soon)
- Run the simulator
- Last and future evolutions: summary
- Available products and visualisation tools

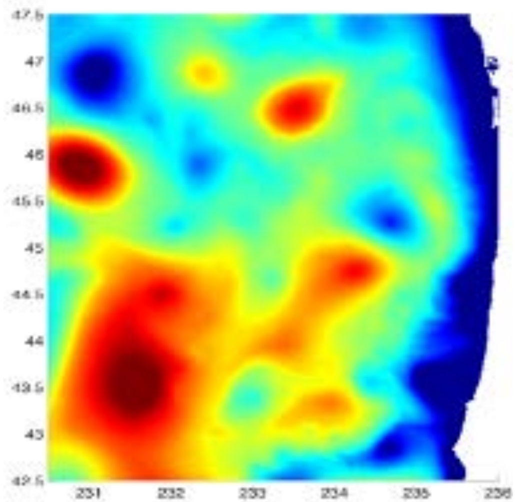
SWOT simulator for Ocean Science

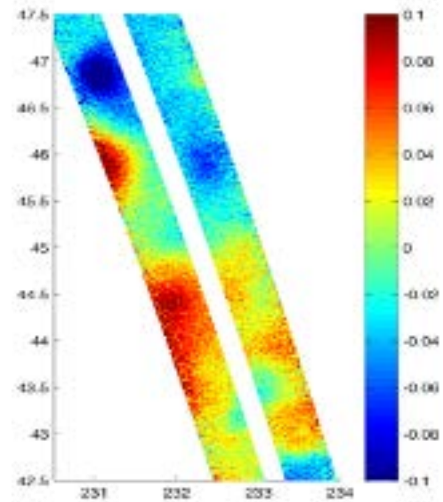
- The needs for a light/portable tool to easily simulate SWOT L2 data with realistic sampling and errors/noise has been pointed by the SDT team.

- **The tool relies on spectral error budget specifications from the project team**
  → It is NOT an instrument simulator.

- Coded in python 3, open source, available on github and conda

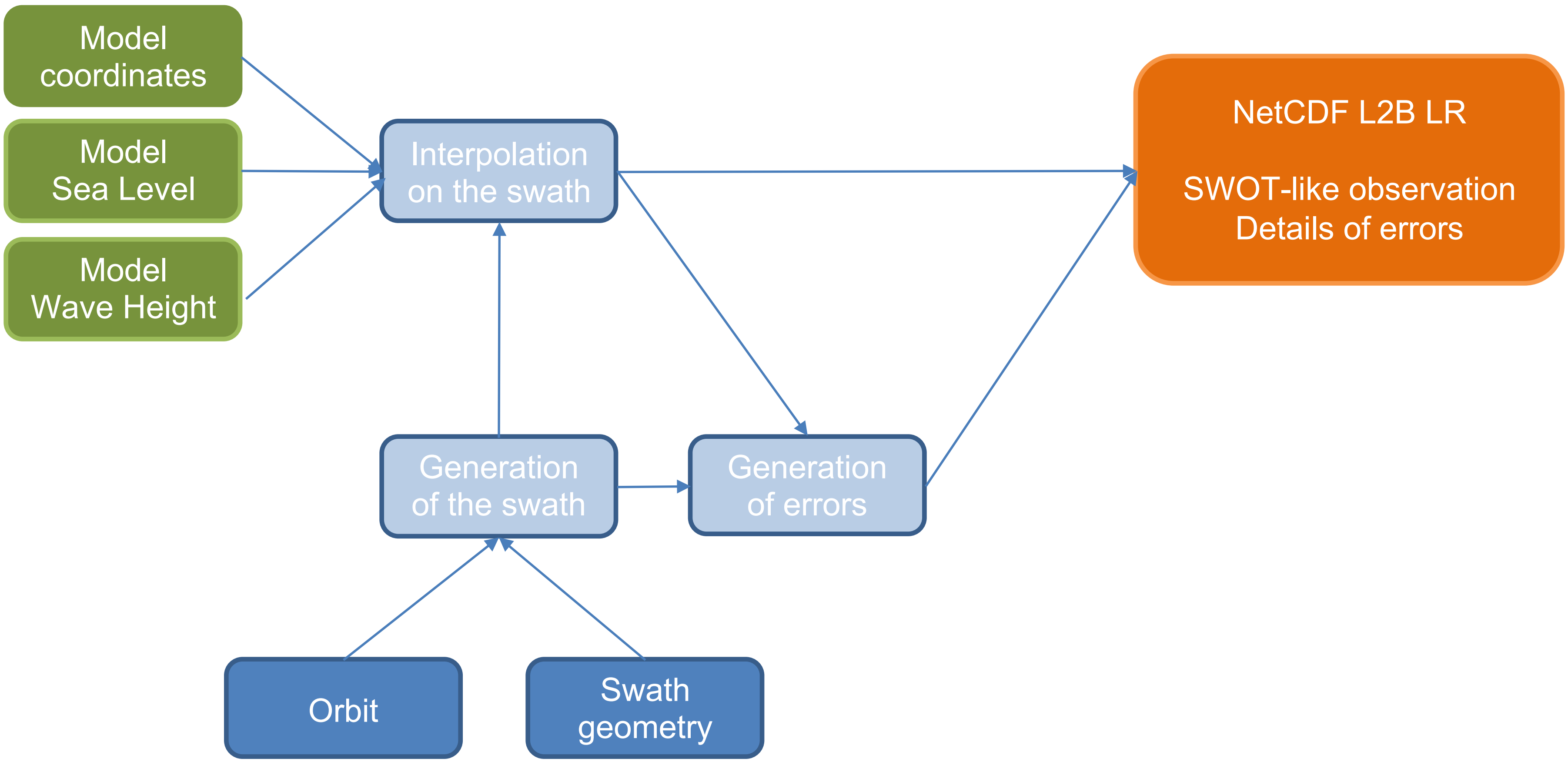
INPUTS: user's model Sea Surface Height Coordinates





OUTPUTS:
SWOT synthetic data sampled on a swath grid
Each error is saved separately

SWOT simulator for Ocean Science



SWOT simulator for Ocean Science

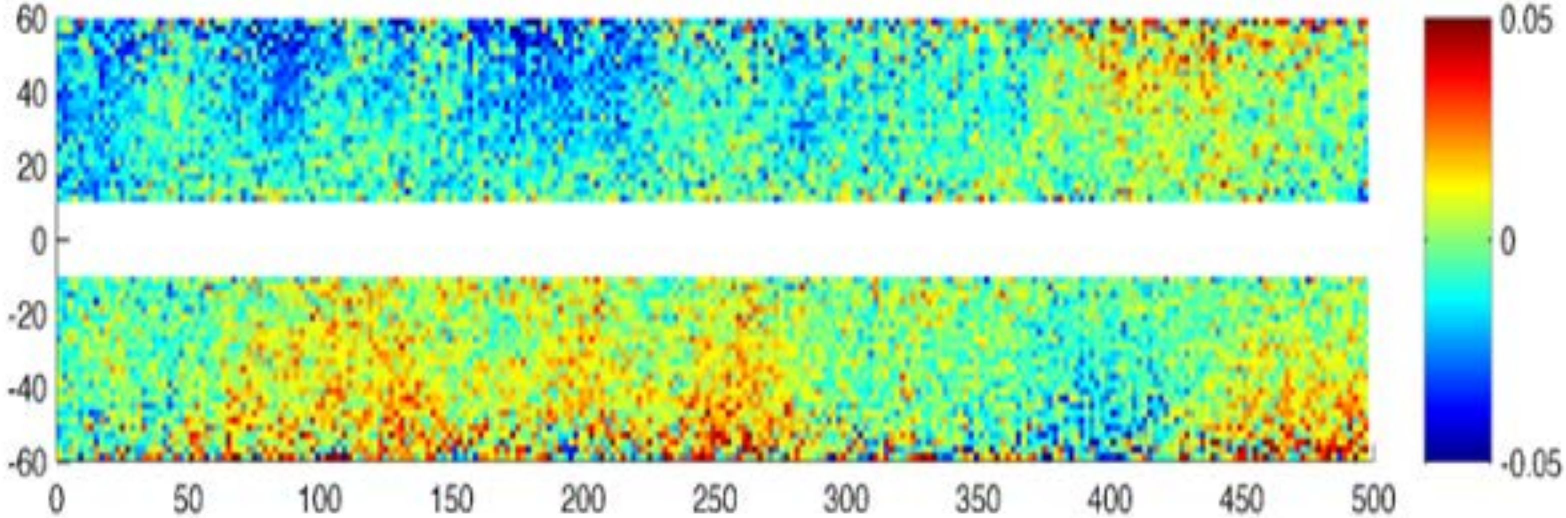
- Open source, use git version, conda version or download tar on the github
- Two versions are available:

	Original version	New version
Git	https://github.com/SWOTsimulator/swotsimulator.git	https://github.com/CNES/swot_simulator
Dependencies	numpy / scipy, netCDF4	numpy / scipy, netCDF4 + pangeo-pyinterp, dask, xarray
Installation	Easy using pip	Can be difficult, use conda
parallelization	Python multiprocessing	Dask parallelization, faster with big files
Input	Most netcdf files, specify format in parameter file	Adapt plugin to convert model file into xarray object to use pangeo-pyinterp
Output	Expert and light netcdf files	Data format similar to one of the future L2B LR SWOT data, nadir format is GPR
Pros	Simple to install Easier to adapt to your own data	Run on HAL efficiently Run on big dataset

➡ **As simulators are evolving, contact us if any question or doubt**

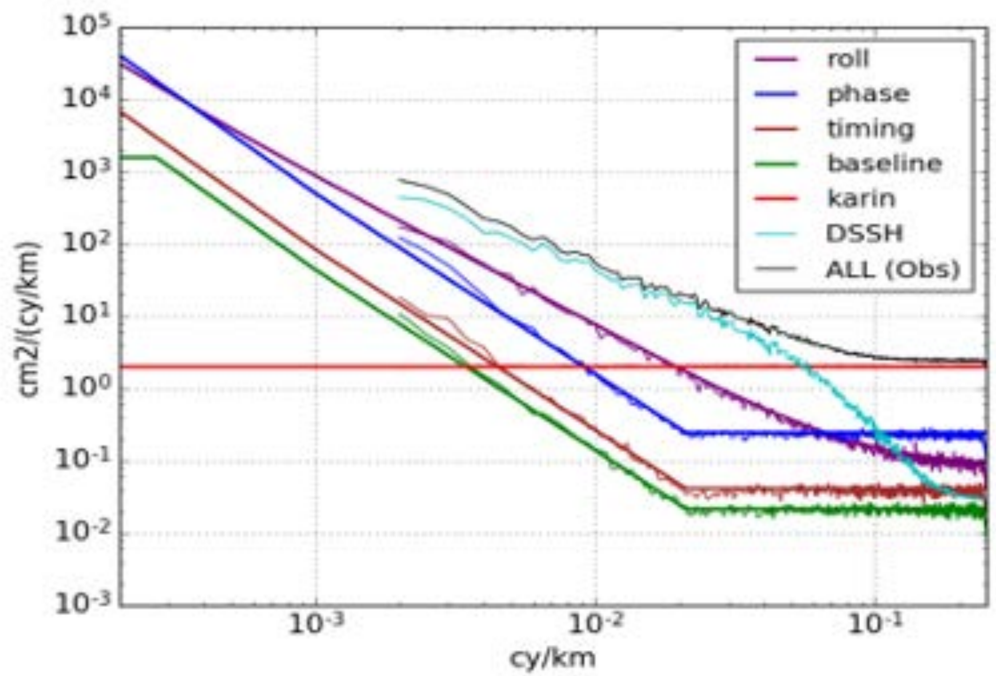
The "Baseline" error simulation

Baseline error realizations

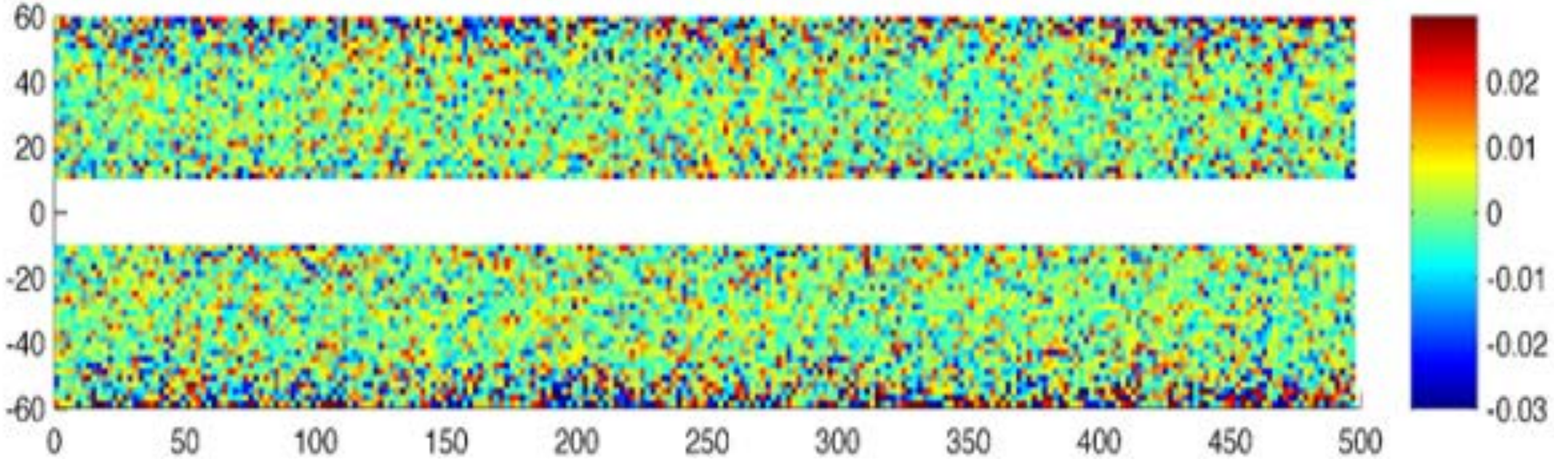


Total error

Random realizations of the noise and errors are performed following the statistical descriptions of the SWOT error budget document [Esteban-Fernandez et al., 2014]

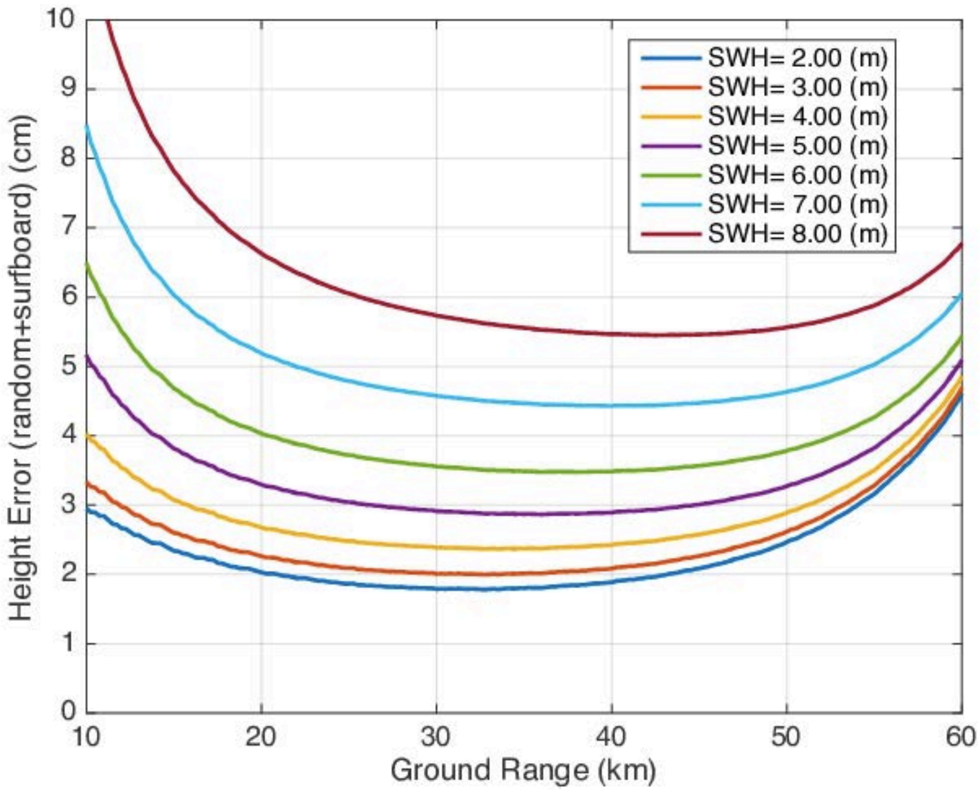


KaRIN noise



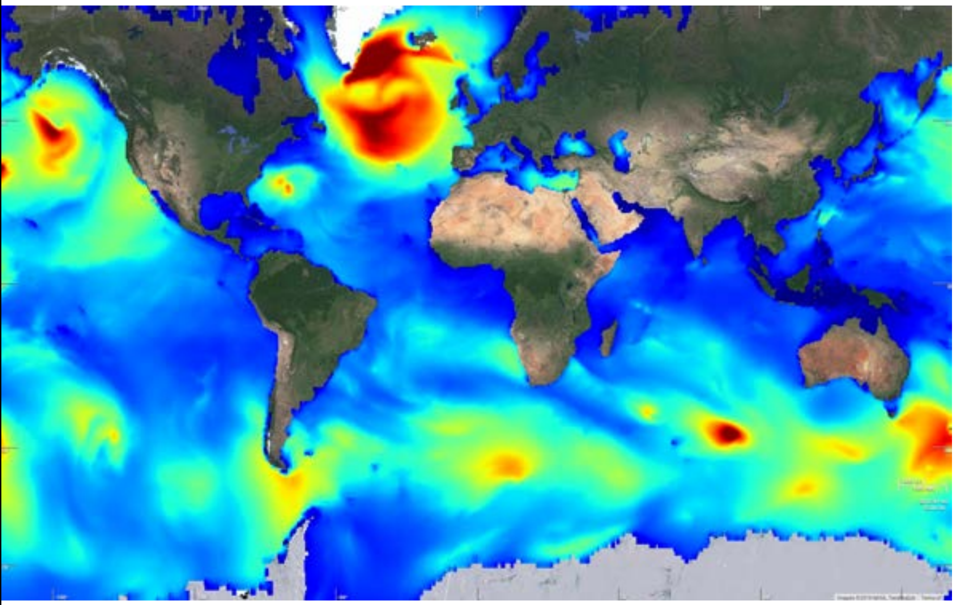
Karin noise

The KaRIN noise is random from cell to cell.
The standard deviation follows the smiley face curves.

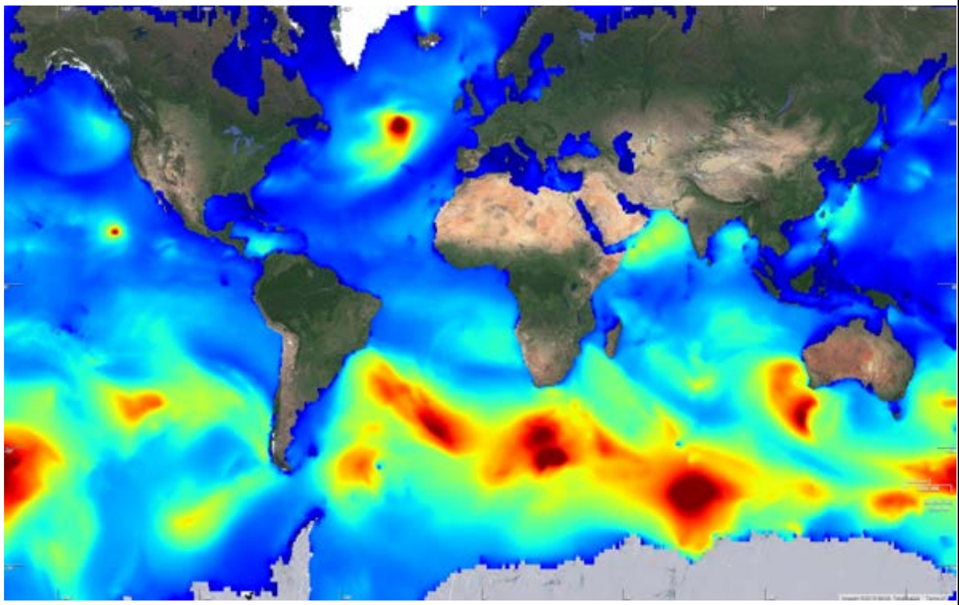


SWH varying in time and space

Karin noise varying with space and time for global simulation

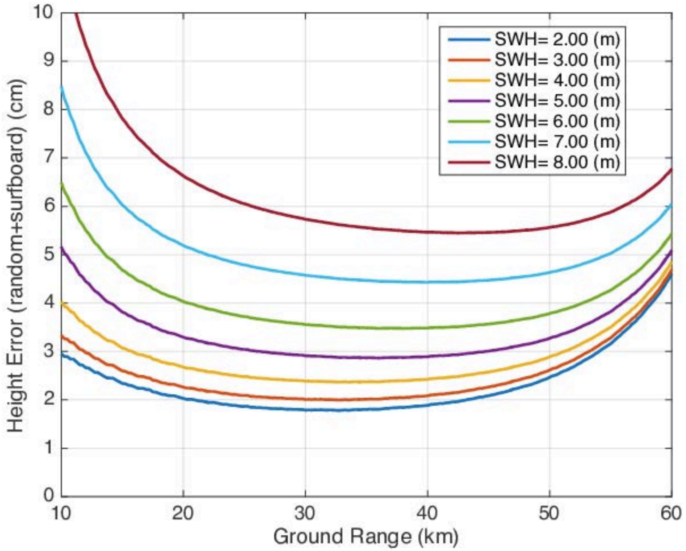


SWH from WW3, mid-January 2016



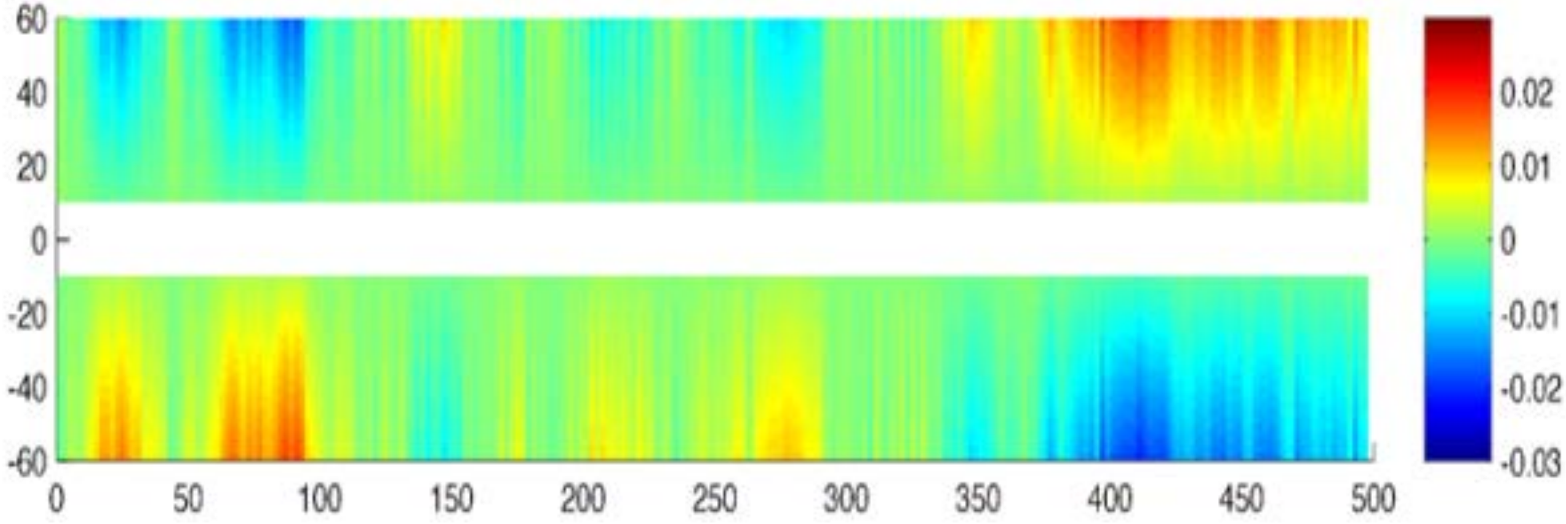
SWH from WW3, mid-July 2016

Capability to read WW3 swh outputs and interpolate it on SWOT grid:
-> Play with SSB correction
-> SWOT-like data available with swh consistent with ssh



No <u>bias</u> due to SWH yet (only noise)

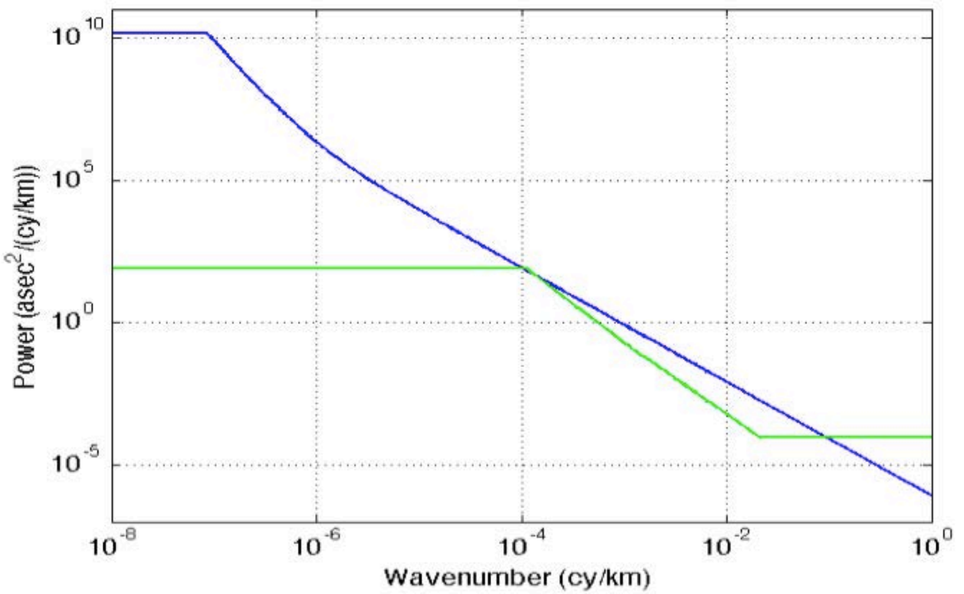
Roll errors



Roll errors

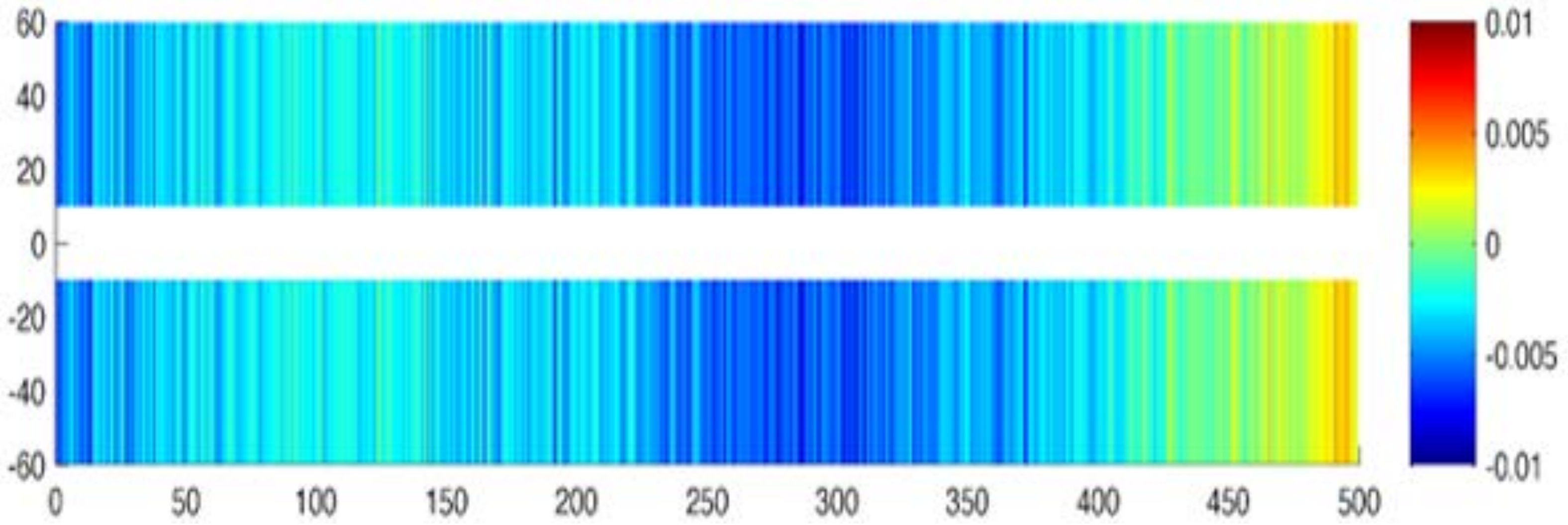
Gyro + knowledge roll errors follow this power spectrum. Errors are projected in the across track direction:

$$h_{roll}(al,ac) = (1+\frac{H}{Re})\theta_{roll}(al)\frac{\pi}{648}ac$$



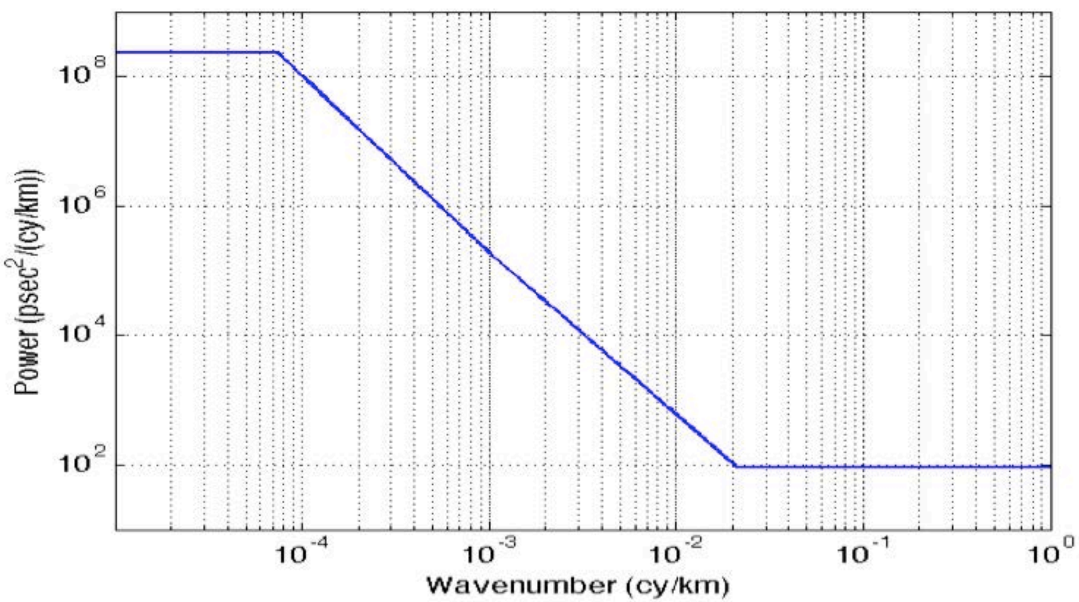
New : faster signal generation

Timing errors

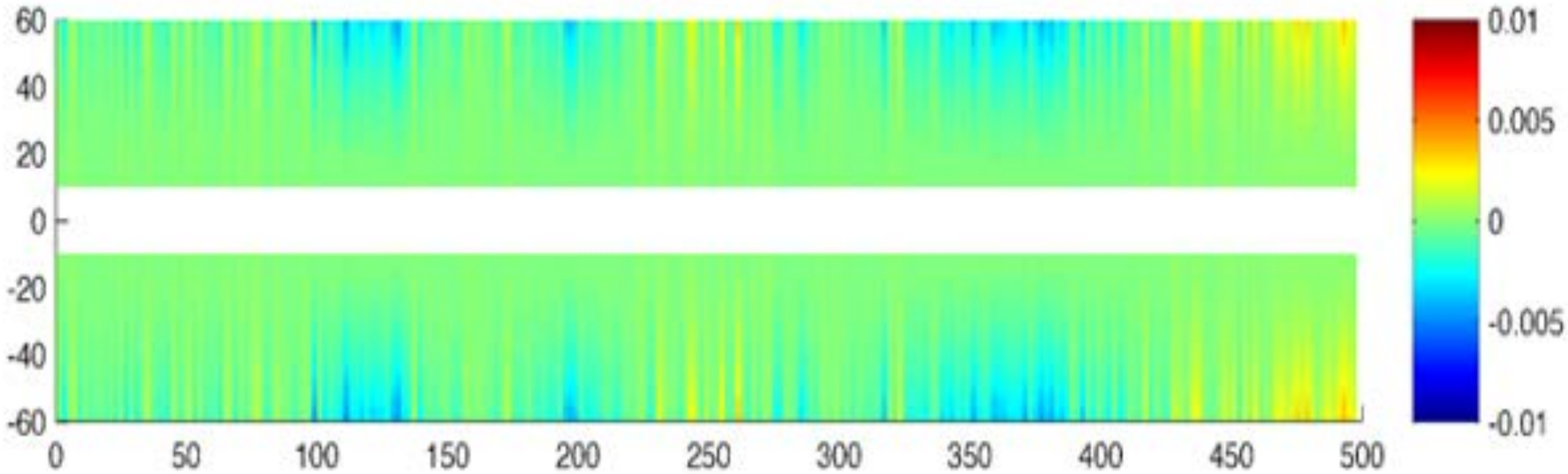


Timing errors

Timing errors follow this power spectrum.



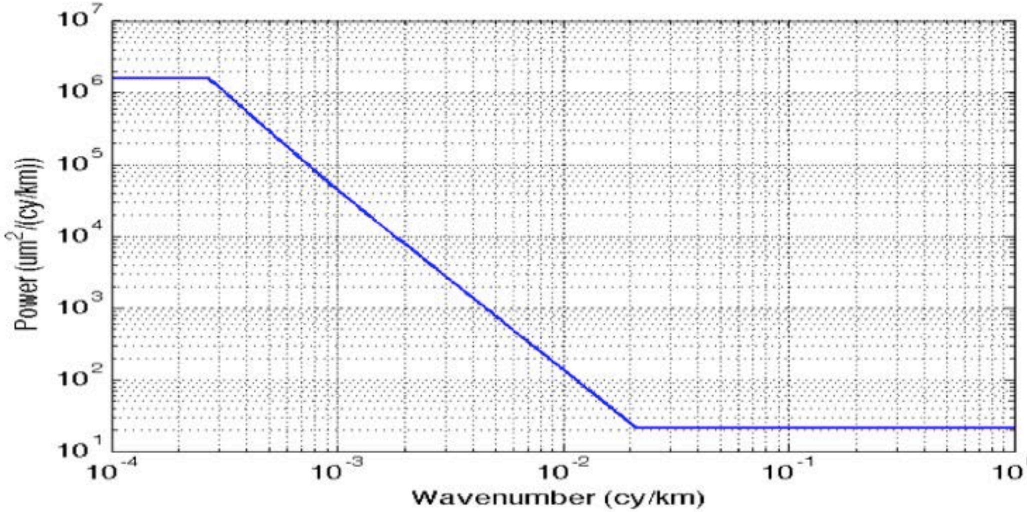
Baseline dilation errors



Baseline dilation errors

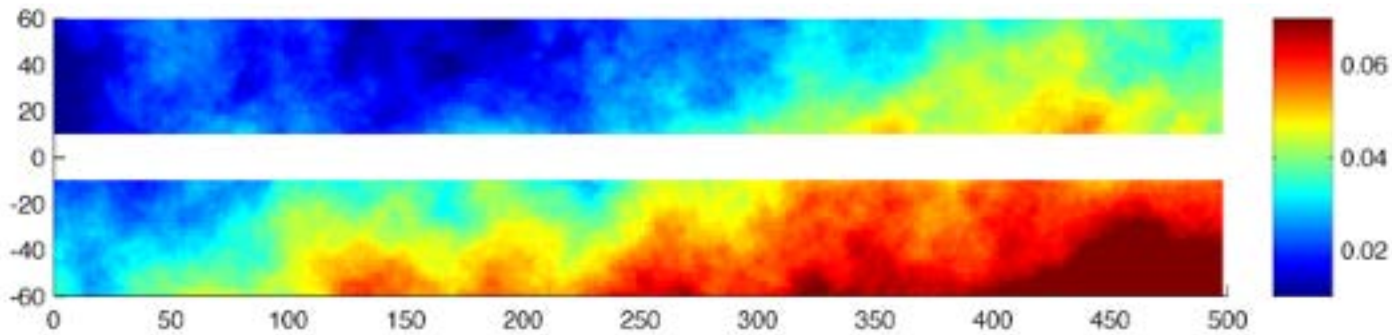
Baseline dilation errors follow this power spectrum. Errors are projected in the across track direction:

$$h_{\delta B}(al, ac) = -(1 + \frac{H}{Re})\frac{\delta B(al)}{HB}ac^2$$



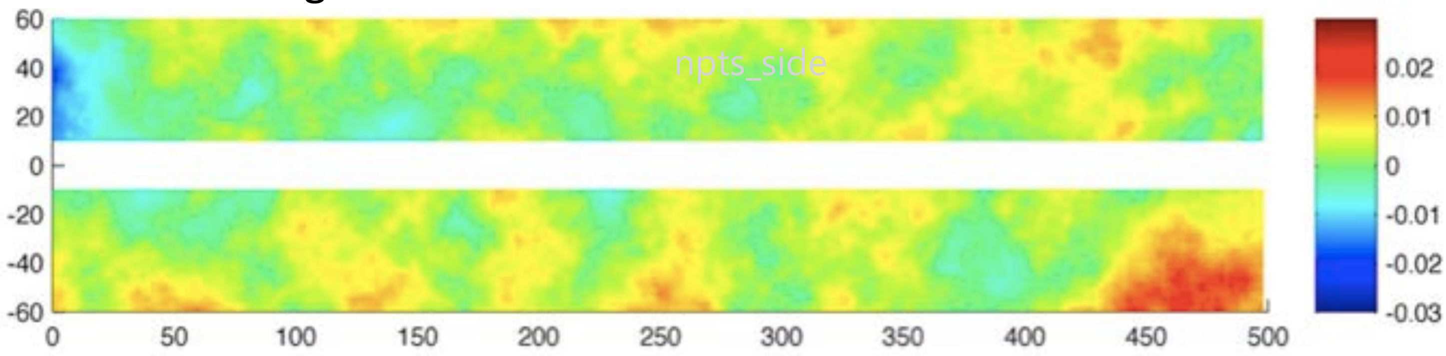
Wet-tropo errors

Generation of wet tropo signal following spectrum

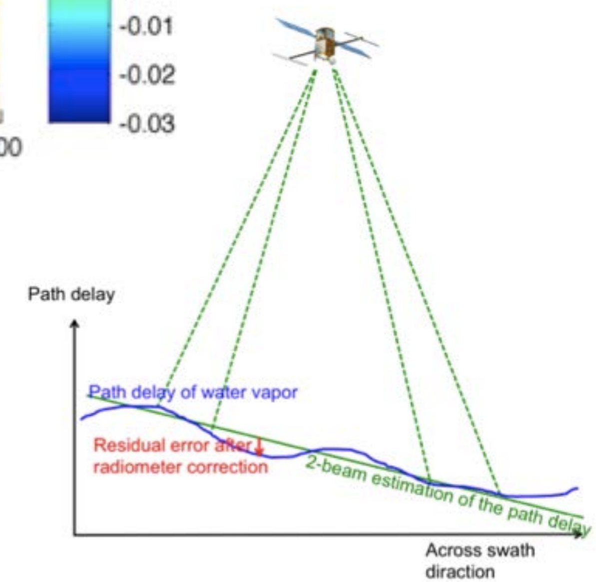


New : MUCH faster signal generation in 2D (based on inverse 2D FFT)

Correction using two beams

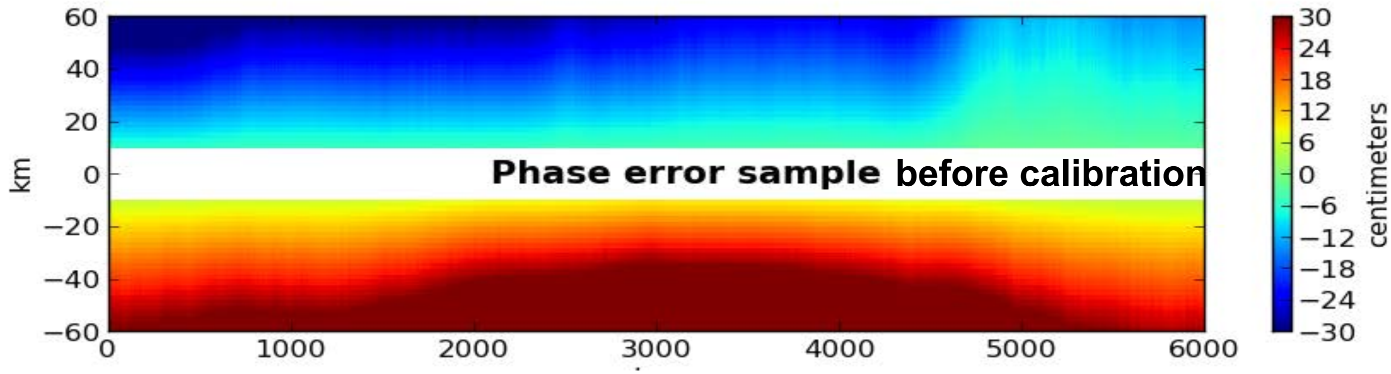


- 8km sigma gaussian footprint size
- Isotropic



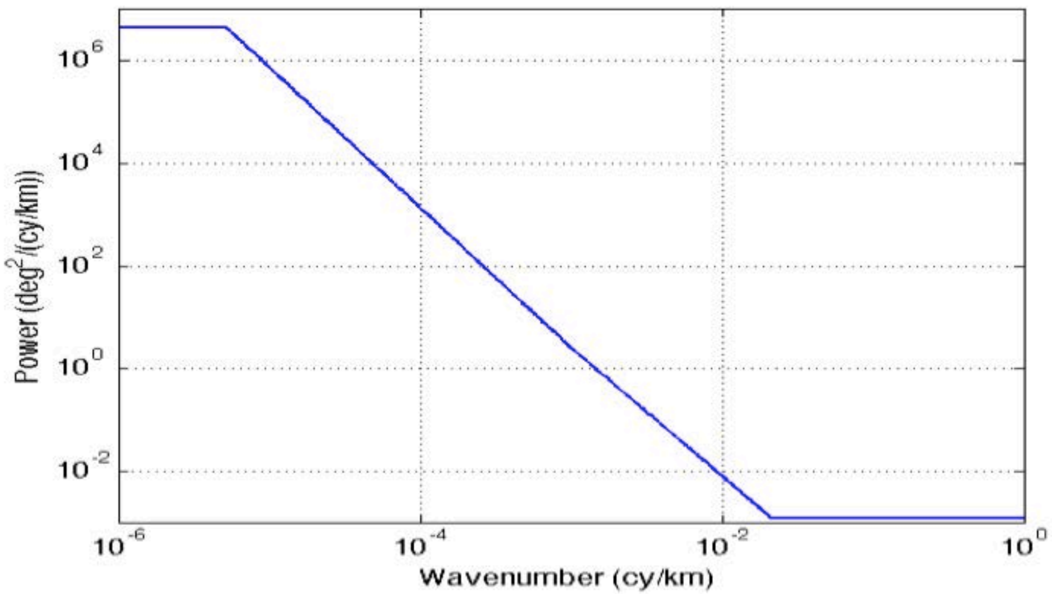
➡ **Wet tropo retrieval algorithm to be updated (2014 version)?**

Phase errors

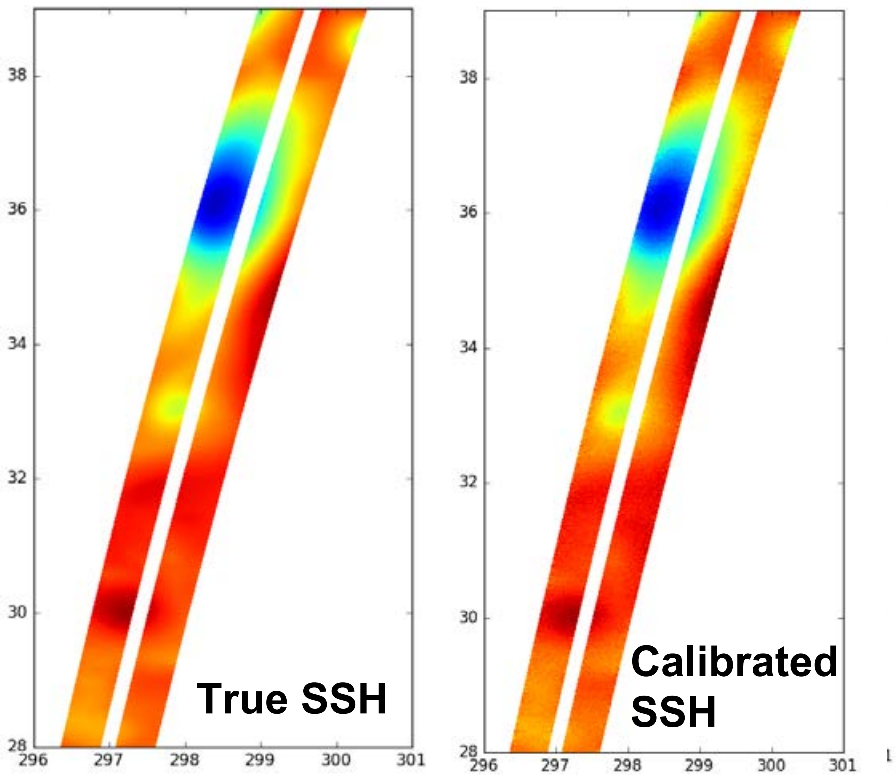


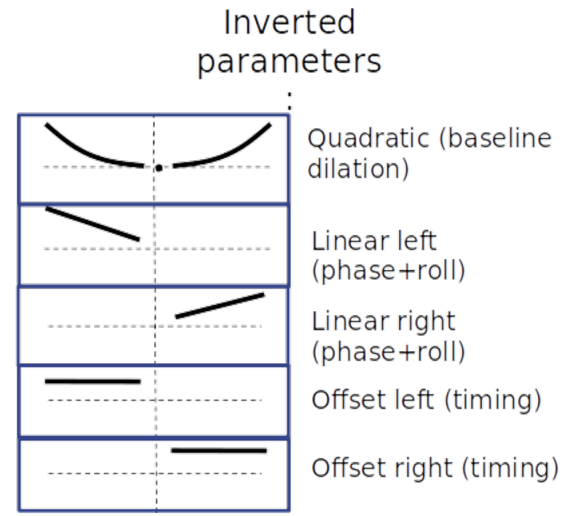
Uncalibrated Phase errors follow this power spectrum. Errors are projected in the across track direction:

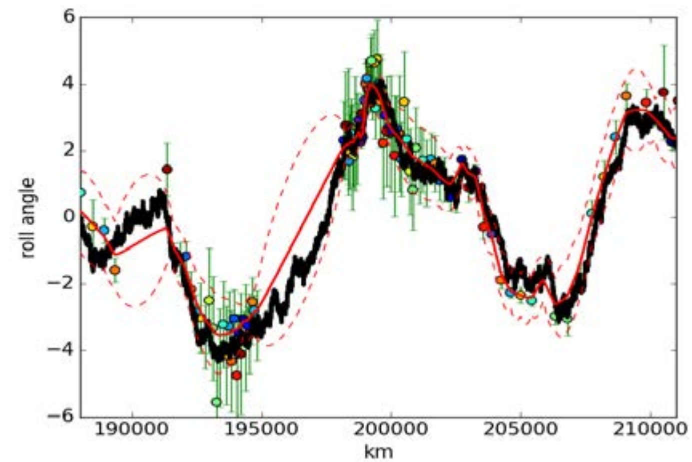
$$h_\theta(al, ac) = \frac{1}{K_{Ka}B}(1 + \frac{H}{Re})\theta(al)\frac{100\pi}{18}ac$$



Simulated crossover calibration







- The systematic errors, phase in particular, feature strong biases (compliant with requirements)
- **Operational crossover calibration** for hydrology can be applied on the Ocean to correct for **long-wavelength error (>2000km).**

The "Current Best Estimate" error simulation

Ongoing implementation of the CBEs

- So far, all input errors were derived from the Baseline requirement spectra

- More 'optimistic' input errors are now available from the SWOT project (from thermal simulations, …) for **Roll knowledge, Phase, baseline dilation, timing errors**

- **Roll gyro error** still derived from the Gyro baseline spectrum

- Addition of **orbital perturbations :** stable (>50 orbits) harmonics on roll at orbital frequency.

--> These new errors are overall much lower at short scales (<1000km) with strong biases that disappear after Xover calibration : **the residual errors (after calibration simulation) will be provided (Oct. 2021)**

Run the simulator

Fill the parameter file:

```python
from os.path import expanduser
import os
home = expanduser("~") + '/src/'
# ------ Directory that contains orbit file:
dir_setup = os.path.join(home, 'swotsimulator', 'data')
# ------ Directory that contains your own inputs:
indatadir = os.path.join(home, 'swotsimulator', 'example',
                         'input_fields')
# ------ Directory that contains your outputs:
outdatadir = os.path.join(home, 'swotsimulator', 'example',
                          '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 = "OREGON"
#Number of processors to be used
proc_number = 1
# ------ Deactivate printing of progress bar to avoid huge log
progress_bar = True

# -----------------------#
# 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 = True
# ------ 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
# -----------------------#
```

> swotsimulator params.py

```python
import os
import swot_simulator.plugins.ssh
import xarray as xr


# Geographical area to simulate defined by the minimum and maximum corner
# point :lon_min, lat_min, lon_max, lat_max
#
# Default: None equivalent to the area covering the Earth: -180, -90, 180, 90
area = [0, -60, 20, -50]

# Distance, in km, between two points along track direction.
delta_al = 2.0

# Distance, in km, between two points across track direction.
delta_ac = 2.0

# Ephemeris file to read containing the satellite's orbit.
ephemeris = os.path.join("/mnt/data/data_swot", "ephem_science_sept2015_ell.txt")

# Index of columns to read in the ephemeris file containing, respectively,
# longitude in degrees, latitude in degrees and the number of seconds elapsed
# since the start of the orbit.
# Default: 1, 2, 0
ephemeris_cols = [1, 2, 0]

# If true, the swath, in the final dataset, will contain a center pixel
# divided in half by the reference ground track.
central_pixel = True

# If true, the generated netCDF file will be the complete product compliant
# with SWOT's Product Description Document (PDD), otherwise only the calculated
# variables will be written to the netCDF file.
complete_product = True

# Distance, in km, between the nadir and the beginning of the swath
half_gap = 10.0

# Distance, in km, between the nadir and the end of the swath
half_swath = 70.0
# Limits of SWOT swath requirements. Measurements outside the span will be set
# with fill values.
requirement_bounds = [10, 60]
# The next two parameters (cycle_duration and height) can be read from the
# ephemeris file if it includes these values in comments. The ephemeris
# delivered with this software contain this type of declaration

# Duration of a cycle.
cycle_duration = 20.86455

# Satellite altitude (m)
height = 891000

# True to generate Nadir products
nadir = True

# True to generate swath products
swath = True
```

> swot_simulator --first-date '2019-08-02T12:00:00' --last-date '2019-08-06T00:00:00' params.py

Run the simulator

- Consider the provided orbits for the two phases of the mission:

Orbit	Repeat cycle (days)	Number of passes
Fast Sampling orbit	0.99349	28
Science orbit	20.8646	584

- Possibility to unplug the operational crossover calibration and implement you own calibration (look at data with or without cross calibration)

- Possibility to simulate other altimetric observations (e.g. Jason, AltiKa, Sentinel, …): OSSEs with a constellation of nadir altimeters

Last / future evolutions: summary

- Major features implemented the past two years:

- SWH varying in time and space for Karin Noise
- Roll-phase crossover calibration
- Orbital perturbation

- Major Bugs corrected

- Swath was stored from right to left instead of left to right in both simulators (since 2014 ... corrected few days ago)
- Randomness was the same from swath to swath in the new version (corrected in May)

- To be implemented this year:

- All instrumental errors to be corrected by crossover calibration
- Best estimate error scenarios

- Still to be implemented:

➡ **Errors due to SSB**

Available data on HAL

Data are being reprocessed currently as the matrix were filled from right to left

SWOT-like data available on HAL:

- Interpolated from SSH AVISO field, and from the updates made by GECO, on the various static and dynamic geophysical corrections for the month of September 2016:

- SWOT-like data: ssh_karin, ssha_karin, swh_karin (from mfwam model without any errors)
- Flag, geolocation: distance_to_coast, ancillary_surface_classification_flag, dynamic_ice_flag
- Ancillary data: mss, geoid, mdt, depth, tides, dac, dry_trop, wet_tropo, ice concentration
- Samples also availble at https://podaac.jpl.nasa.gov/SWOT?tab=datasets&discipline=ocean&sections=about

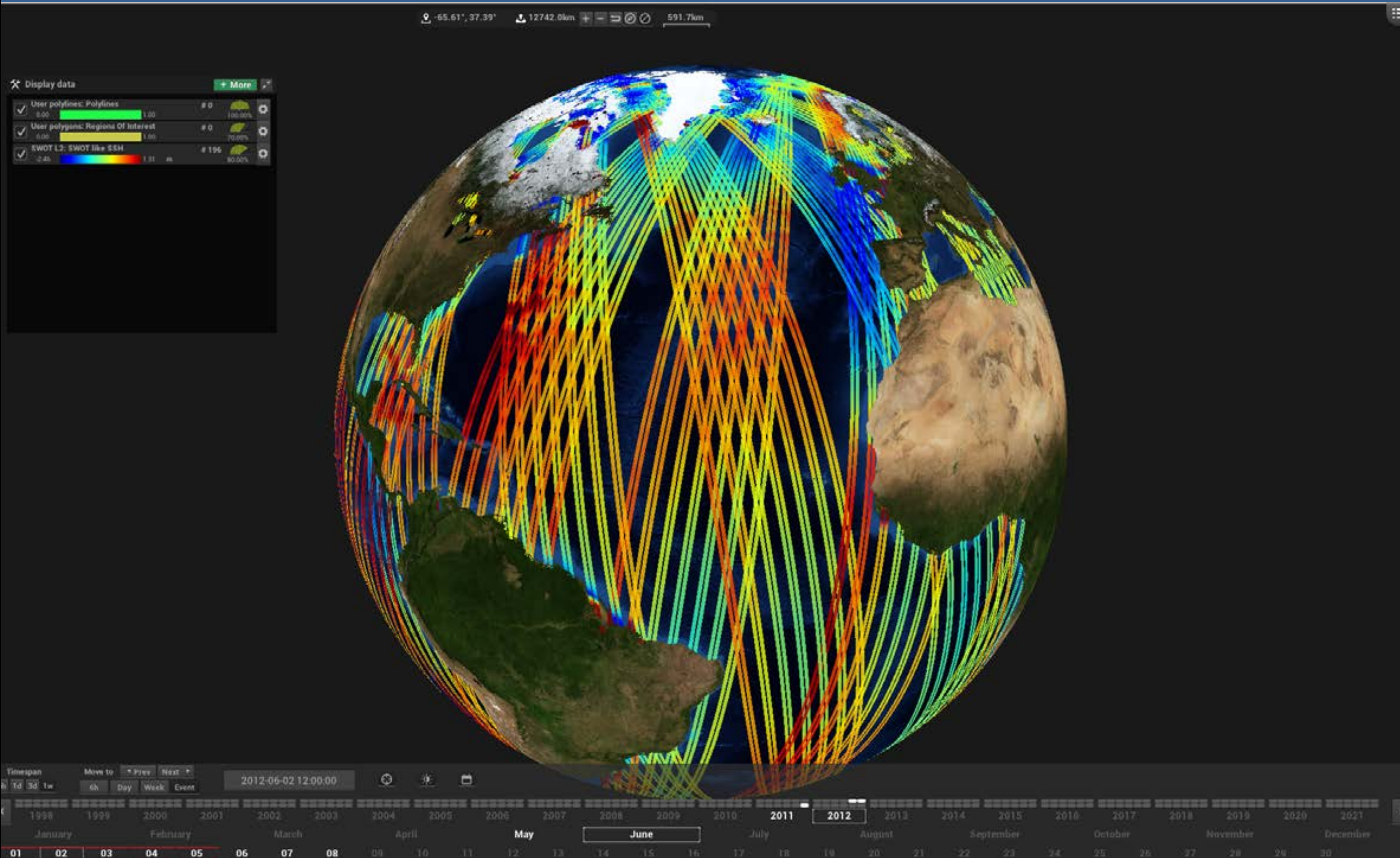
- Interpolated from the MITGCM LLC4320 model for one year of data. Both CalVal and Science phase have been simulated:

- SWOT-like data: ssh_karin
- Flag, geolocation: distance_to_coast, ancillary_surface_classification_flag, cross_track_distance
- Ancillary data: mss, geoid, depth, tide (from FES)
- Simulated errors: roll, phase, karin, timing, wet_troposphere, baseline_dilation
- True SSH

- Interpolated from the GLORYS run with WW3 SWH: to be ready soon

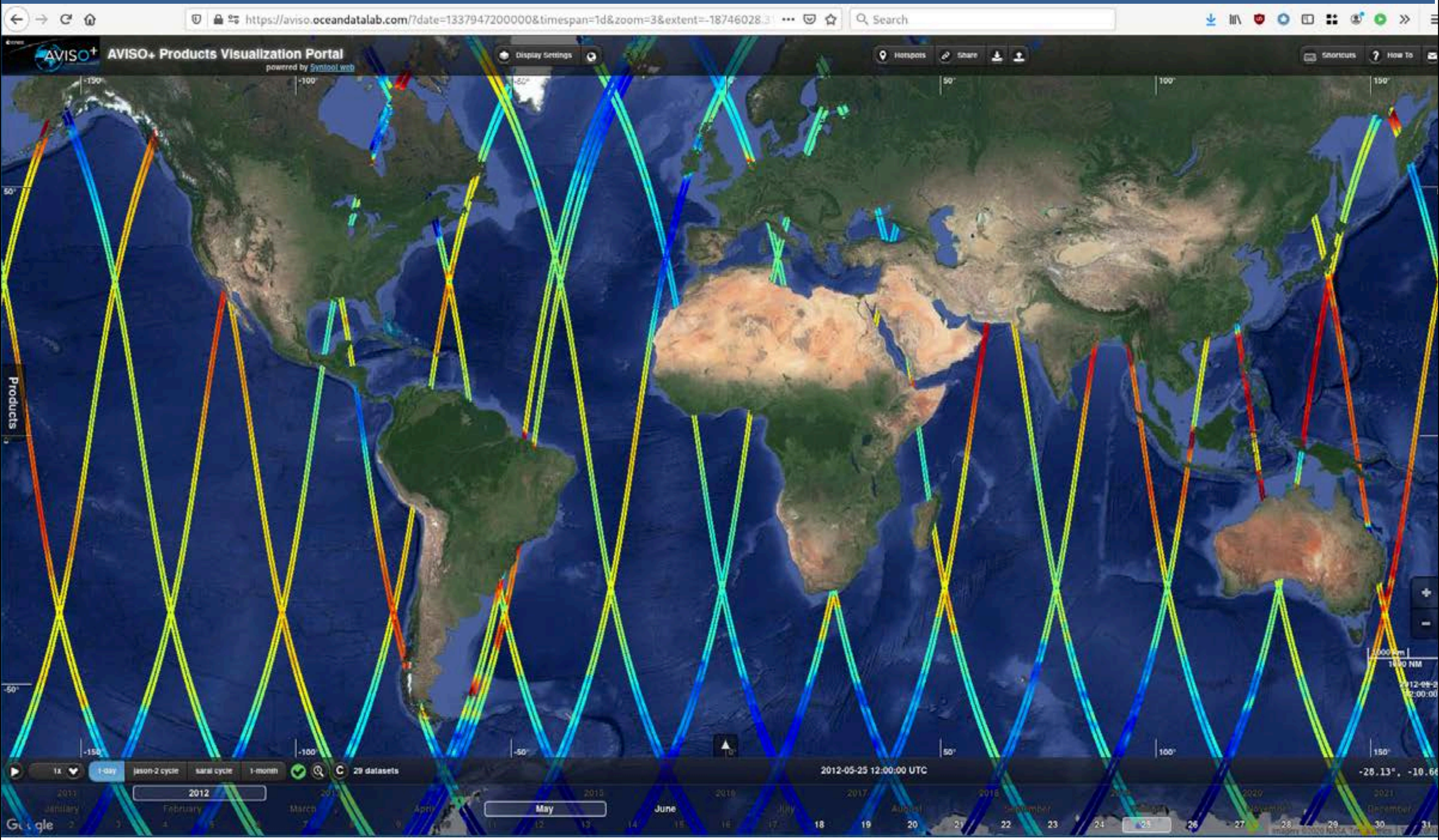
- SWOT-like data: ssh_karin
- Flag, geolocation: distance_to_coast, ancillary_surface_classification_flag, cross_track_distance
- Ancillary data: mss, geoid, depth
- Simulated errors: roll, phase, karin, timing, wet_troposphere, baseline_dilation
- True SSH
- SWH

SEAScope to visualise and analyse data



- SEAScope is a stand alone tool that collocates data in time and space on a 3D globe
- Data can be easily exported in Jupyter notebooks

Available data on web portal



https://aviso.oceandatalab.com