SKIM Simulator Documentation

Release 3.0.0

Lucile Gaultier

Sep 17, 2019

CONTENTS

CHAPTER

ONE

SKIM SIMULATOR FOR OCEAN CURRENT

1.1 Description

This software simulates SKIM sea surface current observations that can be applied to an ocean general circulation model (OGCM).

1.2 Licence

1.3 Installation

The code runs with python and uses the following libraries:

- Numpy
- Scipy
- NetCDF4 if you need to read netcdf4 model files (not included in Canopy)

If you don't have python and the needed python libraries on your machine you can download the enthought python distribution canopy at <https://store.enthought.com/> and follow the installation instructions.

To install skimulator:

-> If you have installed canopy:

Open a canopy terminal (`Tools - Canopy Terminal`)

```
> cd [yourpath]/skimulator/
> python setup.py install
```

->If you have your own python and libraries:

- global installation (requires root privilege):

```
> sudo python setup.py install
```

- local installation:

```
> python setup.py install --home=your/local/path
```

- Uninstall:

```
> sudo python setup.py install --record record_files.txt
> cat record_files.txt | xargs sudo rm -rf
```

1.4 Orbits data

Reference orbits data are available online:

```
$ cd [yourpath]/skimulator/data
$ wget "https://oceandatalab.com/files/skim/orbits.tar.xz"
$ tar xJf orbits.tar.xz
```

1.5 Rain data

Rain statistical rain flags are available on the ftp for the Gulf Stream Region and the Equator:

```
$ cd [yourpath]/skimulator/data
$ wget "https://oceandatalab.com/files/skim/rain.tar.xz"
$ tar xJf rain.tar.xz
```

1.6 Running

Run the SKIM simulator:

For l2b products:

```
> skimul2b [your params file]
```

For l2c products:

```
> skimul2c [your params file]
```

For l2d products:

```
> skimul2d [your params file]
```

1.7 Documentation

- To build the documentation, in the `doc` directory:
  - Build html: `make html`
  - Build pdf: `make latexpdf`

  The build documentation files are located in `doc/build/html` and in `doc/build/latex/`
- for a complete description: see the `doc` directory or just run `pydoc PyDom`

CHAPTER

TWO

SKIM SIMULATOR

Lucile Gaultier (OceanDataLab)

2.1 Abstract:

This software simulates sea surface radial current (Level-2) synthetic observations of the proposed SKIM mission that can be applied to an ocean general circulation model (OGCM), allowing the exploration of ideas and methods to optimize information retrieval from the SKIM Mission in the future. From OGCM currents and Stoke drift inputs, the software generates SKIM-like outputs on a rotating geometry around the orbit ground track, as well as outputs from a nadir altimeter. Some measurement error and noise are generated according to technical characteristics published by the SKIM project team. Not designed to directly simulate the payload instrument performance, this SKIM simulator aims at providing statistically realistic outputs for the science community with a simple software package released as an open source in Python. The software is scalable and designed to support future evolution of orbital parameters, error budget estimates from the project team and suggestions from the science community.

2.2 Simulation of the SKIM sampling over synthetic Sea Surface current

From a global or regional OGCM configuration, the software generates radial velocities, including instrumental and geophysical noise, for each beam. Note that for an accurate instrumental and geophysical noise, various forcings are needed such as mean square slope, Stockes drift, wind, ice ...

2.2.1 Proposed SKIM orbits

The software uses as an input the ground-tracks of the satellite orbit.

	Repeat Cycle (days)	Repeat Cycle (Orbits)	Sub-cycles (days)	Inclination	Elevation (km)
sentinel 1	12	175	6	90.18	698
metop	**29**	**412**	**5**	**98.63**	**817**
fast sampl	3	43	0	98.5	775
fast sampl scanning	8	113	0	98.8	845

The ground-track coordinates corresponding to these orbits are given as input ASCII files of 3 columns (longitude, latitude, time) for one complete cycle sampled at every ~5 km. The first ascending node has been arbitrarily set to 270 degree of longitude, but the user can shift the orbit by any value in longitude. The default orbit is metop.

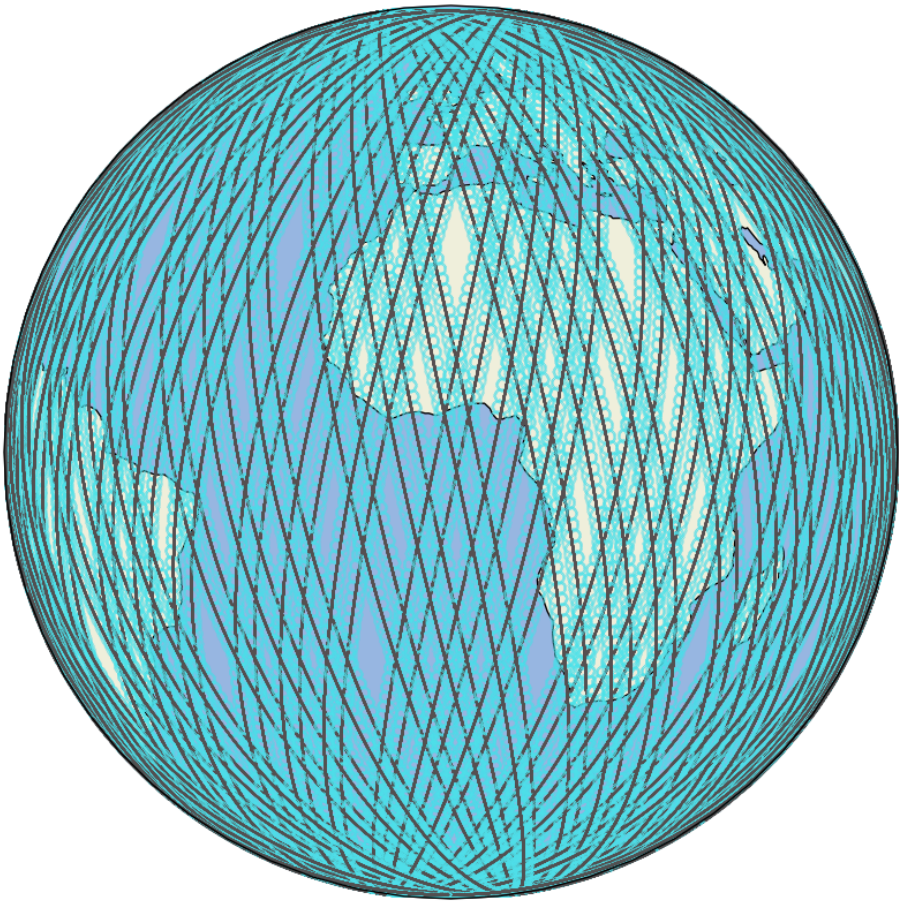

Fig. 1: FIG. 1: 5-day worth of SKIM simulated data in a global configuration with themetop orbit.

Other orbit files of the same format (time, longitude, latitude) can also be used as an input. To avoid distortions in the grid, we recommend a minimum of 10km sampling between the ground-track points of the orbit.

Note that the first two commented lines of the files concerns the satellite cycle (in days) and elevation (in km).

```
cycle = 29
elevation = 817000
```

If these lines does not exist, the skimulator will look for these values in the parameter file or take default value (cycle = 29 days and elevation = 817000)

2.2.2 The SKIM geometry

From the orbit nadir ground track the software generates a grid covering the swath over one satellite cycle. The longitude and latitude coordinates as well as the time are referenced for each grid point. A scheme of the SKIM geometry is presented on *Fig. 2*. The SKIM grid is stored by pass (e.g. 412 ascending passes and 412 descending passes for the Metop orbit). A pass is defined by an orbit starting at the lowest latitude for ascending track and at the highest latitude for descending track. The first pass starts at the first lowest latitude crossing in the input file, meaning that ascending passes are odd numbers and descending passes are even numbers.

2.2.3 Interpolation of model variables on the SKIM grid and nadir track

A list of model variables should be given to the skimulator, as well as a list grids if the coordinates differ from one variable to another. The naming of the netcdf files should be $[pattern\_model]\_[pattern\_variable].nc$, where $pattern\_variable$ is a string. All input variables must be given at the same regular time step.

The absolute time of the first time step is zero and corresponds to the beginning of pass 1. A first date can be provided in order to have a consistent timestamps in the netcdf file. All provided variables are interpolated on the SKIM grid and nadir track for each pass and successive cycles if the input data exceeds 1 cycle. Current and Stokes drift are then projected along the radial component as only measurement along the radial axis can be made.

No interpolation is made in time (the model file with the closest time step is chosen). This avoids contaminations of the rapid signals (e.g. internal waves) if they are under-sampled in the model outputs. However, note that locally, sharp transitions of the variable along the swath may occur if the satellite happens to be over the domain at the time of transition between two time steps. By default a linear 2D spatial interpolation is performed to compute the variable data on the SKIM grid.

Fig. 3a shows an input current as an example. *Fig 3b* is the interpolated and radial component of the current.

2.3 Simulation of errors

2.3.1 Instrumental errors

The instrumental error corresponds to the geometric doppler. This component is proportional to σ^0 and follows the azimuthal dependance

provided by the industry for a σ^0 of 0.66 dB for a 12^o, and 0.99 dB for a 6^o.

A multiplicative coefficient is also applied if the cycle length is smaller than 0.0368 The following variables are needed to compute long range and short range mss: mssu, mssc, mssd, uwnd, vwnd, ucur, vcur.

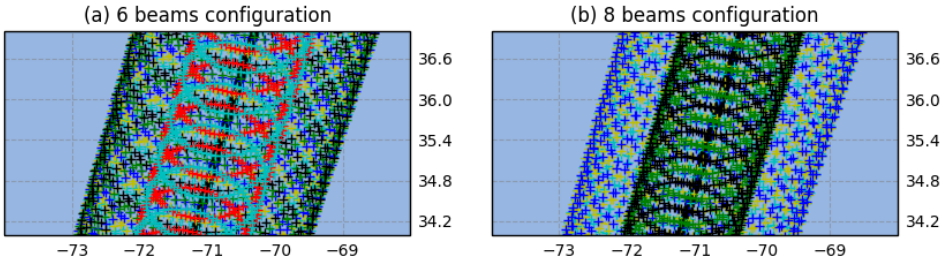

Fig. 2: FIG. 2: scheme of the SKIM geometry with 3 beams at 12 degrees and 2 beams at 6 degree for figure a and 5 beams at 12 degrees and 2 beams at 6 degree for figure b.

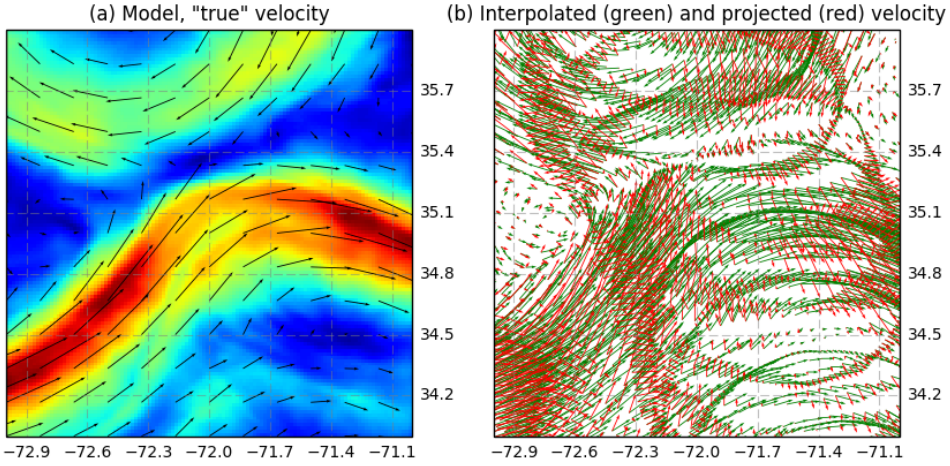

Fig. 3: FIG. 3: Model interpolated currents and the corresponding radial currents.

Computation of long range MSS:

$$mssxl = mssu * \cos(mssd)^2 + mssc * \sin(mssd)^2$$
$$mssyl = mssu * \sin(mssd)^2 + mssc * \cos(mssd)^2$$
$$mssxyl = (mssu - mssc) * \frac{\sin(2 * mssd)}{2}$$

Computation of short range MSS:

$$nwr = \sqrt{(uwnd - ucur)^2 + (vwnd - vcur)^2}$$
$$wrd = \pi/2 - arctan2(vwnd - vcur,\ uwnd - ucur)$$
$$mssshort = \log(nwr + 0.7) * 0.009$$
$$mssshort[mssshort < 0] = 0$$

Directionality for short wave mss (if 0.5: isotrophic)

$$facssdw = 0.6$$
$$mssds = facssdw * mssshort$$
$$msscs = mssshort - mssds$$
$$mssxs = msscs * \sin(wrd)^2 + mssds * \cos(wrd)^2$$
$$mssys = mssds * \sin(wrd)^2 + msscs * \cos(wrd)^2$$
$$mssxys = |mssds - msscs| * \sin(2 * wrd)$$

Computation of total MSS:

$$mssx = mssxs + mssxl$$
$$mssy = mssys + mssyl$$
$$mssxy = mssxys + mssxyl$$

σ^0 on water is computed from the total MSS:

$$B = -0.5 * \tan(beam)^2 * \frac{(\cos(azimuth)^2 * mssy + \sin(azimuth)^2 * mssx - \sin(2 * azimuth) * mssxy)}{mssx * mssy}$$

$$A = \frac{R^2}{(2 * \cos(beam)^4 * \sqrt{mssx * mssy}}$$
$$\sigma^0_{water} = A \exp(B)$$

with $R^2 = 0.55$ which is a typical value for the tropics in Ka band. Note that R depends on the radar frequency, water temperature and salinity (e.g. $R^2 = 0.50$ for 3ºC water).

In the presence of ice, we use the concentration of sea ice C_{ice} and assume that σ^0_{ice} is constant ($\sigma^0_{ice} = 2.5$ for 6º beam and $\sigma^0_{ice} = 1$ for 12º beam).

$$\sigma^0 = (1 - C_{ice}) * \sigma^0_{water} + C_{ice} * `\sigma^0_{ice}$$

Finally, the distribution of the instrumental error is a function of the azimuth following curves provided by the instruments simulator and a stretching proportional to σ^0 is applied. A random number is picked from this distribution.

2.3.2 Wave Doppler

The geophysical doppler includes also part of the currents due to the Stokes drift. This component is later referred as the current wave Doppler Uwd. To compute the wave doppler, a parametrisation has been learned. It depends

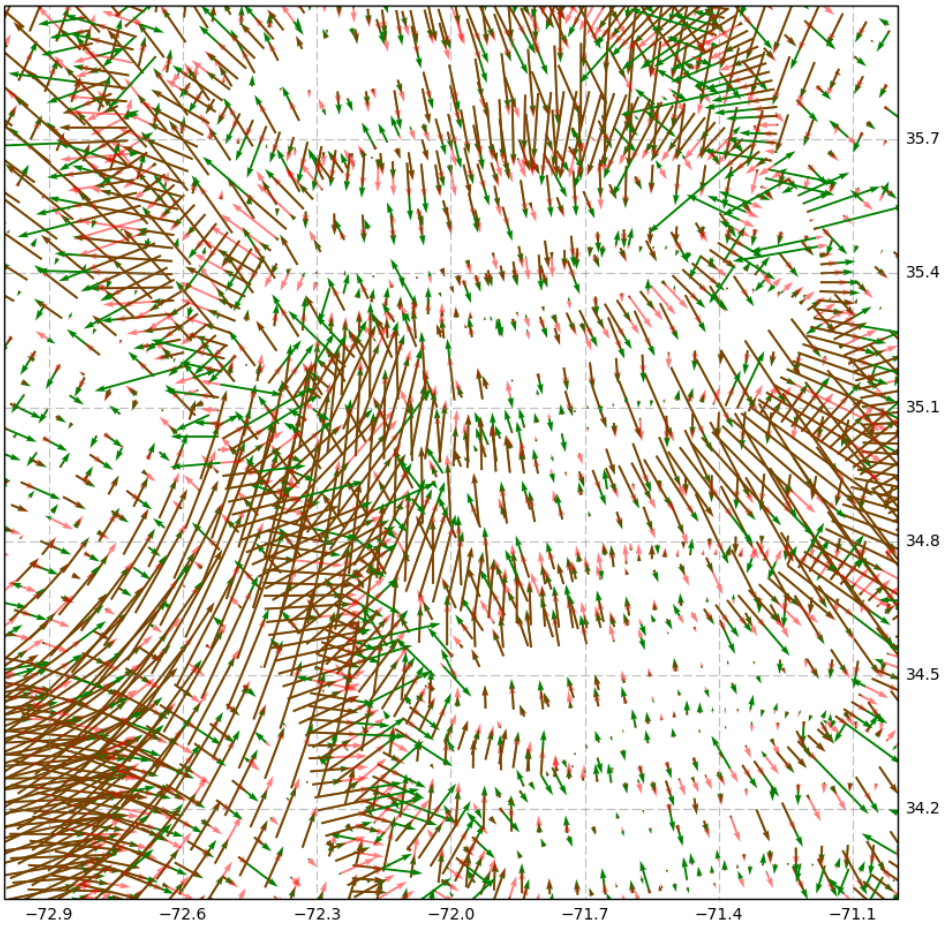

Fig. 4: FIG. 4: Model interpolated currents and the corresponding instrumental error.

on the radial Stokes drift ($ussr$), the norm and the radial wind (:math *nwnd*, :math‘wndr‘), the mean square slope (mss) and the wave height (hs). These parameters are either interpolated from the model (true value) or we simulate their estimation from a wave spectrum. Note that the along track spectrum is supposed to be very noisy and thus an azimuthal interpolation is performed to assess along track signal.

The retrieval of the radial Stokes drift from a wave spectrum is estimated by averaging all the projected Stokes on the azimuth in a radius of 70 km. As there is a 180 degree ambiguity, the wind direction is considered to determine the $ussr$ sign. The wave height is retrieved from the altimeter, thus the nearest point at nadir is considered. The mean square slopes retrieval from the spectrum is simulated by either using the nearest point at 6º / nadir or by averaging all available noisy components in a 70 km radius. Note that a random noise has been applied to the mss signal.

The parametrisation has been learned.

The ‘true’ wave doppler is computed using the previous parameters interpolated from the model inputs. The ‘estimated’ wave doppler is computed using the ‘estimated’ parameters as explained in the previous paragraph. The error on the wave doppler is the difference between the ‘estimated’ and the ‘true’ doppler.

The estimation of the Stokes drift and the mean square slope is degraded near the coast where all the azimuth are not available, and in areas where there is an important gradient. This is also the case in marginal ice zone, where points where there is ice are detected an handled separately.

In the marginal ice zone, open ocean and areas with ice are treated separtely to retrieve the Wave Doppler as the mss varies a lot between those two types of water .. _Fig5:

2.3.3 σ^0 gradient in footprint

This gradient is statistically computed by stretching pdf of gradient of σ^0 as a function of sigma0 derived from Saral-Altika altimeter (in band Ka).

The pdf of Sarah-Altika has been provided by CNES.

The gradient of sigma0 is then converted into a horizontal velocity.

2.3.4 Rain flagging

For each region, an ensemble of scenes is provided to statistically simulate rain patterns consistent with the weather in the area. These scenes have been derived from GPM observation at 5 km resolution. As the rain patterns are not correlated from on day to another, for each pass, a scene is randomly picked and interpolated on the swath as a function of the latitude. A rain flagging is applied by masking all points higher than a given threshold.

Note that there is also the possibility to derive the rain from model files if they are provided in the list of variables to be interpolated. Then this variable will be used for rain flagging.

2.3.5 Atmospheric gradient

A change in water content in the atmosphere have an impact on σ^0. The corresponding sigma0 gradient is computed directly from the gradient of PIA from the scenes previously used for rain flagging (derived from GPM data). The sigma0 gradient is then converted into a horizontal velocity

2.3.6 Attitude

AOCS remaining error is simulated using a spectrum provided by the technical instrumental team, thermal elastic dilatation are parametrised using tables provided by instrumental simulation. Both of these error are converted into

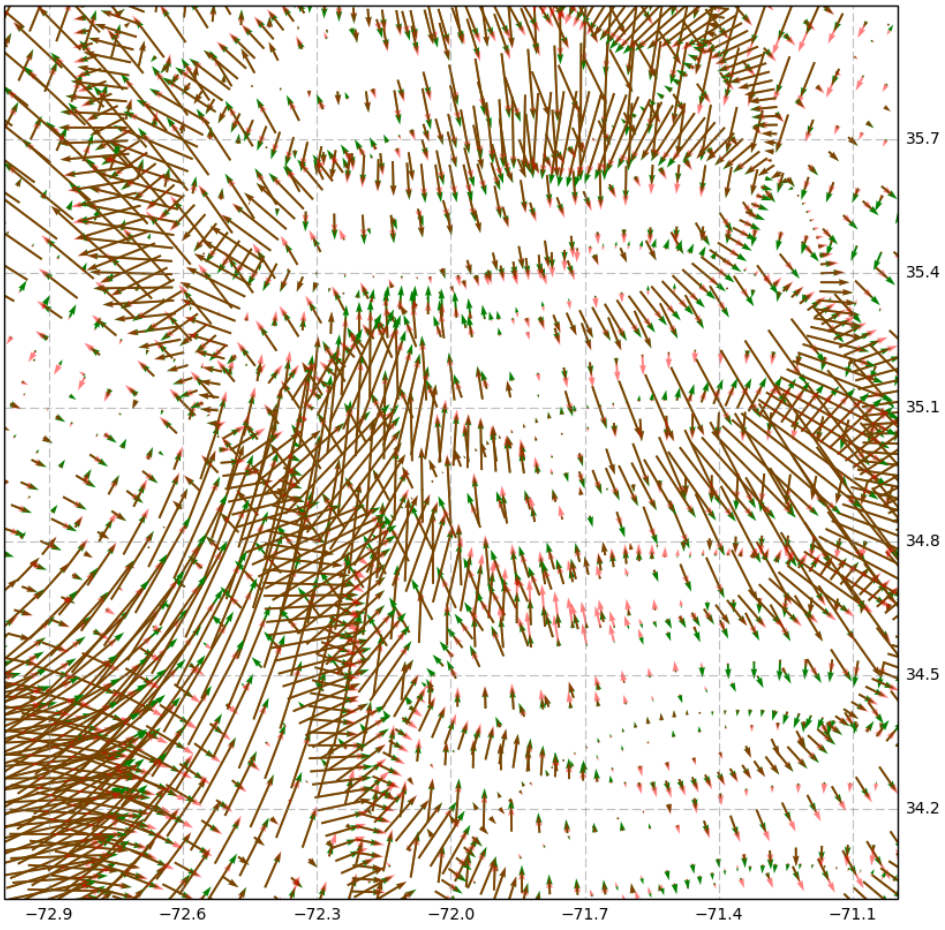

Fig. 5: FIG. 5: Model interpolated currents and the corresponding wave remaining doppler.

velocities. Their correction is performed offline and the remaining error can be read into a file and added to the error budget.

2.3.7 Total error

All previous errors are added to compute the total error.

2.4 Simulation of errors for the nadir altimeter

Two main components of the nadir altimetry error budget are simulated: the altimeter noise and residual wet-tropospheric errors. For the altimeter noise, the noise follow a spectrum of error consistent with global estimates from the Jason-2 altimeter. The wet-tropospheric residual errors (not implemented yet) are generated using the simulated wet-tropospheric signal and radiometer beam convolution described in SWOT Simulator documentation.

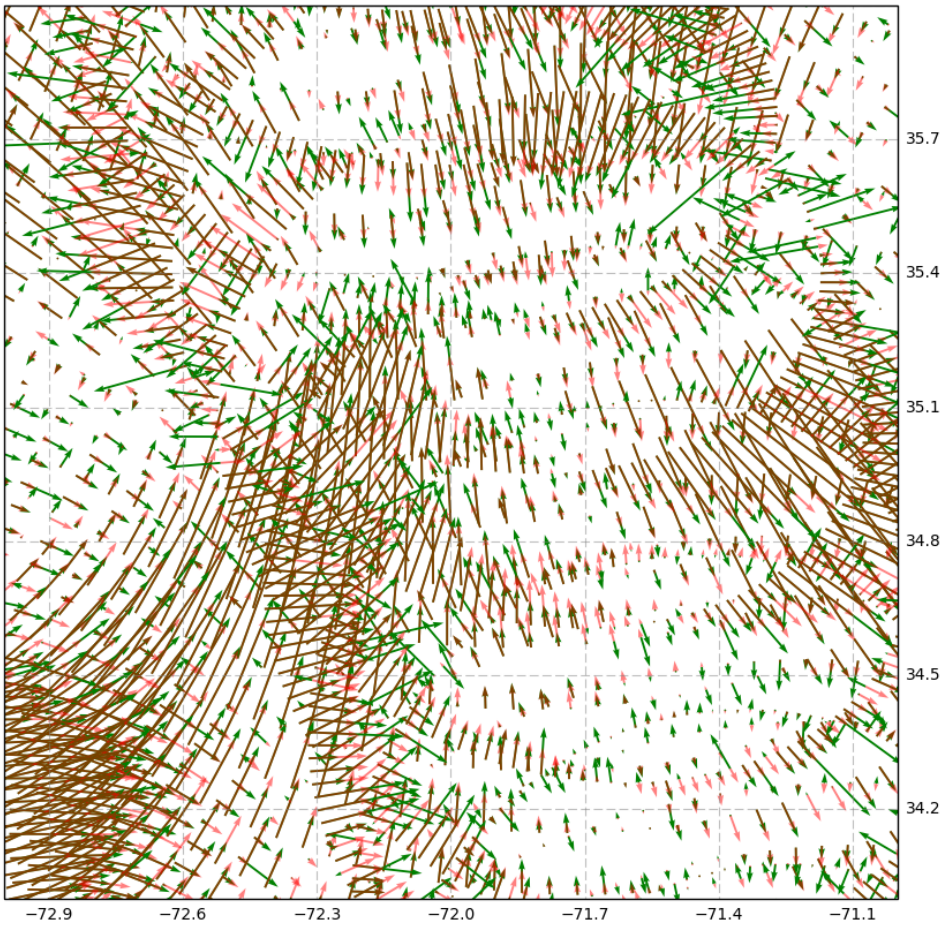

Fig. 6: FIG. 6: Model interpolated currents and the corresponding total noise (instrumental + geophysical).

2.5 The software

The software is written in Python, and uses Numpy, Scipy and netCDF4 python libraries. Pyresample is also required for L2C computation and faster L2B interpolation. All the parameters that can be modified by the user are read in a params file (e.g. params.py) specified by the user. These parameters are written in *yellow* later on and are linked to their location in the params file example.

The software is divided in 11 modules:

- `run_simulator.py` is the main program that runs the simulator.
- `mod_run.py` contains interpolations and data construction functions.
- `build_swath.py` generates the SKIM geometry and save several coordinates and angular variables in a netcdf file.
- `build_error.py` generates all the errors on the swath.
- `rw_data.py` contains all the classes to read and write model and SKIM data (in netcdf).
- `mod_tools.py` contains miscellaneous functions (algebraic functions and generation of random coefficients).
- `regridding.py` contains reconstruction algorithms for L2c products.
- `regridding_l2d.py` contains reconstruction algorithms for L2d products.
- `mod_uwb_corr.py` contains function to estimate wave doppler and associated parameters
- `mod_parallel.py` contains function for parallelisation
- `grid_check.py` is a module that check if grids have change in order to proof any mistake from the user

The directory share contains numpy data needed for Wave Doppler computation and TED componants

2.5.1 Inputs

You can provide to the simulator a list of model outputs in netcdf. You need to have at least the meridional and zonal currents to compute error-free radial L2B velocities and SSH if you want to simulate nadir data. Wind and MSS are necessary to compute instrumental noise (proportional to $sigma^0$), Stokes drift, MSS, wave height and Wind are needed to compute true and estimated wave doppler. Ice concentration should also be provided for realistic estimation of sigma0 in polar areas. Any other variables provided to the skimulator will be interpolated on the SKIM points. If rain is provided in these files, it will be interpolated and used for rain flagging

A list of files (in .txt format) is provided using *file_input* in the parameter file.

The extension should not be provided in the list_of_files:

```
model_0001_
model_0002_
model_0003_
```

The corresponding model file for a variable *var* should be model_0001_var.nc For example, if all the variables are in the same file $myfile\_[date].nc$, the list of file will be:

```
myfile_date1
myfile_date2
myfile_date3
```

FIG 19: Example of a list of files, a list is provided in the example directory.

The grid files are provided as a list in the parameter file, using the key *file_grid_model*. Make a list of all grid files that are necessary for your variables, the link between the variable and the grid is given in a number in the *list_input_var*. If no file_grid_model is provided, The skimulator is going to use the first file of your list and data in this file will be ignored.

It is possible to generate the SKIM sampling alone, without using any model as an input. If the name of the list of files (*file_input*) is set to *None*, then only SKIM grid files will be generated.

The module `rw_data.py` is used to read model data. For any standard netcdf format, the data can be read using *model* = MODEL_NETCDF, which is the *model* default value. The user needs to specify the list of latitude (*lat*) and longitude (*lon*) variables names corresponding to the list of grid files provided in *file_grid_model*. All other variables that are to be read, are added to the dictionary *list_input_var*:

```
list_input_var = {'key': [[variable\_name], [variable\_extension_file], [number\_
↪corresponding\_to_grid_file]]}
```

The following table summarizes the key that are required to compute instrumental noise and wave bias:

Key	corresponding variable	Necessary to compute . . .
ucur	Zonal total current	Wave bias, radial current
vcur	Meridional total current	Wave bias, radial current
uuss	Zonal Stokes drift	Wave bias
vuss	Meridional Stokes drift	Wave bias
ice	Ice concentration	Instrumental noise if there is ice
mssd	Direction long wave mss	Instrumental noise
mssx	Zonal MSS	Instrumental noise
mssy	Meridional MSS	Instrumental noise
ssh	Sea Surface Height	Nadir SSH
uwnd	Zonal wind	Wave bias, Instrumental noise
vwnd	Meridional wind	Wave bias, Instrumental noise
rain	rain quantity in mm/h	Rain flagging if no scene avail.

Netcdf data that follow WW3 format can automatically be read using *model* = WW3 and there is no need to specify the longitude or latitude variables name. Below is an example of *list_input_var* for WW3 model (all variables are on the same grid):

```
file_grid_model = ('grid.nc', )
lon = ('longitude', )
lat = ('latitude', )
list_input_var = {'ucur': ['ucur', 'cur', 0],
                  'vcur': ['vcur', 'cur', 0],
                  'uuss': ['uuss', 'uss', 0],
                  'vuss': ['vuss', 'uss', 0],
                  'ice': ['ice', 'ice', 0],
                  'mssd': ['mssd', 'msd', 0],
                  'mssx': ['mssc', 'mss', 0],
                  'mssy':['mssu', 'mss', 0],
                  'ssh': ['wlv', 'wlv', 0],
                  'uwnd': ['uwnd', 'wnd', 0],
                  'vwnd': ['vwnd', 'wnd', 0],
                  'rain': ['rain', 'rain', 0],}
```

Below is an example of *list_input_var* for a model with an Arakawa grid (type C):

```
file_grid_model = ('grid_u.nc', 'grid_v.nc', 'grid_T.nc')
lon = ('lon_u', 'lon_v', 'lon_t')
```

(continues on next page)

(continued from previous page)

```
lat = ('lat_u', 'lat_v', 'lat_t')
list_input_var = {'ucur': ['u', 'cur', 0],
                  'vcur': ['v', 'cur', 1],
                  'uuss': ['uuss', 'uss', 0],
                  'vuss': ['vuss', 'uss', 1],
                  'ice': ['ice', 'ice', 2],
                  'ssh': ['wlv', 'wlv', 2],
                  'uwnd': ['u10', 'wnd', 0],
                  'vwnd': ['v10', 'wnd', 1]}
```

The coordinates are supposed to be in degrees and current variables in m/s in the program.

To refer timestamp properly in netcdf files, fill in the *first_time* key following *first_time'='yyyy-mm-ddTHH:MM:SSZ'*. By default, *first_time'='2011-11-15T00:00:00Z'*.

If there is a ice_mask varying in time, set *ice_mask* to False to recompute the mask at every cycle.

The number of time in each file should be constant for all the files and specified in the *dim_time* parameter. The time step between two inputs (*timestep*) and the number of steps that have to be processed (nstep) can be modified in the params file. The value corresponding to not a number can be specified in *model_nan*.

2.5.2 Generation of the SKIM geometry

The SKIM grid is generated in the `build_swath.py` module. The orbit file (*filesat*) is located in *dir_setup* and contains longitude, latitude and the corresponding time for each point of the orbit (see section *Proposed SKIM orbits* for more details on the provided orbit). The orbit is interpolated at the cycle duration time resolution. The rotation speed of the antenna is specified in *rotation_speed* in tr/min. The provided value has been computed to keep an integer number of illumination in the macro-cycle. The geometry of beams is provided with lists and give for each beam a position in radian (*list_pos*), an angle on the sensor in degree (*list_angle*), an order for the illumination (*list_shift*) with the macro-cycle starting with the nadir beam. The generation of the SKIM grid can be made on the whole region of the model (*modelbox* = *None*) or on a subdomain (*modelbox* = [lon_min, lon_max, lat_min, lat_max]). To compute the noise alone (*file_input = None*), a *modelbox* has to be provided. If there is no pass on the required domain, the user can specify a shift in longitude (*shift_lon*).

A netcdf file containing SKIM grid information is stored for each pass in the output directory (*outdatadir*) under the name *filesgrid* _p[pass].nc. It contains nadir variables (_nadir) and other beams variables (one beam per column in the same order as in the parameter file) for the following variables: along track and across track distance from the nadir (in km), longitudes (lon) and latitudes (lat), time, inclination at nadir, the number of days in a cycle (cycle) and the distance crossed by the satellite in a cycle (al_cycle). Once the SKIM grid has been created, it is stored in *outdatadir*. As long as the domain (*modelbox* parameter) and the geometry do not change, the grids do not have to be recomputed and *makesgrid* can be set to False. For a more convenient use, in the example the name of the output files are a concatenation of a *config* name and *satname* orbit name.

2.5.3 Radial current and error fields

At each pass, for each cycle, an output netcdf file containing the currents interpolated from the model as well as the interpolated current projected on the radial direction (if *file_input* is not set to *None*) and the different errors are created. The output file names are *file_output* _c[cycle]_p[pass].nc for the swath and *file_output* _c[cycle]_p[pass].nc for the nadir. The SSH is interpolated on the SKIM grid. If the model grid is regular, option *grid* can bet set to *regular* and RectBivariateSpline interpolation from scipy is used. In all cases, *grid* option can be set to *irregular* and pyresample is used for the interpolation if the module is installed. If the grid is irregular and pyresample is not installed, mgriddata from scipy interpolates data is used with either the 'linear' (*interpolation* ='linear') or 'nearest' neighbor (*interpolation* ='nearest') option. In case of large domain, this last solution for the mgriddata, interpolation can be slow and even trigger memory error. The use of the 'nearest' interpolation is necessary to avoid memory error

though the derivative of the current can be significantly altered using this interpolation method. The *list_output* key concerns the list of variables that you want to be store in the netcdf files. The most common key are summarized in the table below, you can add any other key that you want to interpolate on the SKIM grid:

Key	Long name	Required for . . .
ur_true	Radial error free velocity	ur_obs
ur_obs	Radial velocity with errors	
ucur	Meridional true current	ur_true, ur_obs, sigma0,
vcur	Zonal true current	ur_true, ur_obs, sigma0,
uuss	Meridional Stokes drift	ur_obs, uwd, ussr, ussr_est
vuss	Zonal Stokes drift	ur_obs, uwd, ussr, ussr_est
instr	Instrumental error	ur_obs,
radial_angle	Azimuthal angle	all
uwnd	Meridional wind	uwd, uwd_est, ur_obs, instr
vwnd	Zonal wind	uwd, uwd_est, ur_obs, instr
mssx	Meridional MSS	instr, ur_obs
mssy	Zonal MSS	instr, ur_obs
mssxy	Mixed MSS	instr, ur_obs
uwd	True wave doppler	ur_obs
uwd_est	Estimated wave doppler	ur_obs
sigma0	NRCS	instr, uwd, uwd_est
ssh_obs	Sea Surface Height with errors	ssh_obs, nadir
ssh_true	Error free Sea Surface Height	ssh_obs
ussr	True radial stokes drift.	uwd
ussr_est	Reconstructed radial stokes drift	uwd_est
dsigma0	Azimuthal sigma0 gradient.	ur_obs
dsigm-atm	sigma0 gradient due to atmosphere	ur_obs
yaw_aocs	Yaw due to the error in gyro	
yaw_ted	Yaw due to the thermal dilation	
yaw	Total yaw error	
yaw_rem	Remaining yaw after correction	

To compute each error, set the corresponding parameter to True (*instr*, *ice*, *uwb*, *nadir*,

rain, *attitude*).

If *nadir* is True, then *ncomp1d* should be specified to compute random error realisations. If *rain* is True, the *rain_file* that contains the path to the set of scenes and the *rain_threshold* to flag rain above this threshold should be specified.

All the computed errors are saved in the output netcdf file. The observed SSH (SSH_obs) is also computed by adding all the computed errors to the SSH from the model (SSH_model) when model data are provided.

Two errors are considered in the nadir. The first one is the instrument error, which follows the 1d spectrum computed from the current altimeters. You have to set (*nadir*) to True to compute this error.

2.5.4 L2C 2d currents

L2b radial current can be projected on a along swath and across swath grid using the `skimul2c` command with the same parameter file as the one used for the L2b current production. It uses the L2b produced previously as an input. The along track and across track grid resolution is specified in *posting*. By default, the spatial resolution of the grid is 5 km. The L2c reconstruction uses neighbors to project and interpolate the current. The length resolution to select neighbors can be set in *resol*. Coefficient for the OI are decreasing exponentially with the distance to the pixel. As data around nadir are particularly noisy (all radial velocity are parallels), one can mask them by specifying the distance in

km from nadir where data are to be masked *ac_threshold*. The list of variables to be interpolated on the l2c grid is set in *list_input_var_l2c*.

A l2c_config variable can be specified to label a l2c configuration, this variable will be appended in the file name (`[config]_l2c[l2c_config]_c[ncycle]_p[npass].nc`)

The L2c outputs contains along track, across track, meridional and zonal current reconstructed from the error-free and radial velocity with errors. True along track, across track meridional and zonal velocity interpolated from the model inputs are also stored for diagnosis purposes. All variables specified in the *list_input_var_l2c* will be stored in the l2c Netcdf file.

The following variables can be saved in the L2C output files:

Key	Long name	Reconstruction method
ux_noerr	Error-free zonal velocity	OI
uy_noerr	Error-free meridional velocity	OI
ux_obs	Observed zonal velocity	OI
uy_obs	Observed meridional velocity	OI
u_ac_obs	Observed across track velocity	OI
u_al_obs	Observed along track velocity	OI
u_ac_noerr	Error-free across track velocity	OI
u_al_noerr	Error-free along track velocity	OI
ux_true	True zonal velocity	interpolation
uy_true	True meridional velocity	interpolation
u_ac_true	True across track velocity	interpolation
u_al_true	True along track velocity	interpolation
angle	Angle of xac with eastward vector	
uwnd	Zonal wind at 10m	interpolation
vwnd	Meridional wind at 10m	interpolation
rain	Rain	interpolation
x_al	Along track distance from the beginning	
x_ac	Across track distance from nadir	
u_ac_uwd	Across track Wave Doppler velocity	OI
u_al_uwd	Along track Wave Doppler velocity	OI
u_ac_uwdrem	Across track Wave Doppler error	OI
u_al_uwdrem	Along track Wave Doppler error	OI
u_ac_instr	Across track instrument error	OI
u_al_instr	Along track instrument error	OI
u_ac_dsigm	Across track sigma0 gradient error	OI
u_al_dsigm	Along track sigma0 gradient error	OI
u_ac_uss_obs	Across track estimated Stokes drift	OI
u_al_uss_obs	Along track estimated Stokes drift	OI
u_ac_uss_oi	Across track error-free Stokes drift	OI
u_al_uss_oi	Along track error-free Stokes drift	OI
u_ac_uss_tru	Across track true Stokes drift	interpolation
u_al_uss_tru	Along track true Stokes drift	interpolation
u_ac_dsigma	Across track error due to dsigma	interpolation
u_al_dsigma	Along track error due to dsigma	interpolation

2.5.5 L2D 2d currents

L2b radial current can be projected on a longitudinal and latitudinal grid using the `skimul2d` command with the same parameter file as the one used for the L2b current production. It uses the L2b produced with `skimul2b` as an input.

The along track and across track grid resolution is specified in *posting_l2d*. By default, the spatial resolution of the grid is 0.1° x 0.1°. The L2d reconstruction uses spatio-temporal neighbors to project and interpolate the current. The length resolution to select neighbors can be set in *resol_spatial_l2d* in space and *resol_temporal_l2d* in time. These numbers are scaling factors to increase or decrease default values. Coefficient for the OI are decreasing exponentially with the distance and time to the pixel. The time domain to compute L2d can be set in *time_domain* and the spatial domain in *spatial_domain*. The list of variables to be interpolated on the l2d grid is set in *list_input_var_l2d*.

The L2d outputs contains meridional and zonal current reconstructed from the error-free and radial velocity with errors. True meridional and zonal velocity interpolated from the model inputs are also stored for diagnosis purposes. The truth is averaged over the dtime specified in *resol_temporal_l2d = (start_time, end_time, dtime)*

2.5.6 Getting started

All information regarding the installation and running of the software are in the *README* file. An example of a *params.txt* file is given below.

Once you have installed skimulator, you can print help by typing in a python or ipython window:

```
>>>import skimulator.M
>>>help(skimulator.M)
```

with M the name of the module.

To compute L2b products:

```
>>>skimul2b [your_parameter_file]
```

You can compute L2c products after L2b files have been produced, keep the same parameter file and run:

```
>>>skimul2c [your_parameter_file]
```

You can compute L2d products after L2b files have been produced, keep the same parameter file and run:

```
>>>skimul2d [your_parameter_file]
```

Example of Params.txt for SKIM-like data

```
# ------ Name of the configuration (to build output files names)
config = [yourconfig]
# ------ Directory that contains orbit file:
dir_setup = os.path.join([yourpath], 'skimulator', 'data')
# ------ Directory that contains your own inputs:
indatadir = '[yourpath_to_yourdata]/'
# ------ Directory that contains your outputs:
outdatadir = '[yourpath_to_outputs]/'
# ------ Orbit file:
satname = [chosenorbit]
filesat = os.path.join(dir_setup, [chosenorbit])
# ------ Number of days in orbit (optional if specified in orbit file)
satcycle = 29
# ------ Satellite elevation (optional if specified in orbit file)
sat_elev = 817 * 10**3
# ------ Order of columns (lon, lat, time) in orbit file
# (default is (0, 1, 2) with order_orbit_col = None)
order_orbit_col = None
```

(continues on next page)

(continued from previous page)

```
# , dir_setup+os.sep+'orbjason.txt', dir_setup+os.sep+'orbaltika.txt' ]
# ------ Number of processors for parallelisation purposes
proc_number = [number of processor (integer)]
# ------ Deactivate printing of progress bar to avoid huge log
progress_bar = True or False
```

```
# -----------------------#
# SKIM swath parameters
# -----------------------#
# ------ Satellite grid file root name:
#        (Final file name is root_name_[numberofpass].nc)
filesgrid = os.path.join(outdatadir, '{}_grid'.format(config))
or filesgrid = os.path.join(outdatadir, '[your_grid_root_name]')
# ------ Force the computation of the satellite grid:
makesgrid = True or 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 or [yourlon_min, yourlon_max, yourlat_min, yourlat_max]
#------- Rotation speed of the antenna (in tr/min)
rotation_speed = rotation depends on the chosen config
#------- List of position of beams:
list_pos = (0, [angle_in_rad], [angle_in_rad] ...)
#------- List of angle of beams in degrees:
list_angle = ([incidence], [incidence], [incidence] ...)
#------- List of timeshift as regard to nadir for 12 degree beams:
list_shift = (1, 3, 2 ...)
#------- Cycle duration
cycle = 0.0368
# ------ Shift longitude of the orbit file if no pass is in the domain
#        (in degree): Default value is None (no shift)
shift_lon = 0
# ------ 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, [your_list_of_file_name.txt]' or None
# ------ Type of model data:
#        (Optional, default is NETCDF_MODEL and reads netcdf3 and netcdf4 files)
#        (Other option is WW3)
model = 'WW3' or 'NETCDF_MODEL'
# ------ First time of the model
first_time = 'yyyy-mm-ddTHH:MM:SSZ'
# ------ Grid file name
file_grid_model = (os.path.join(indatadir, [yourgridfileu]),
                   os.path.join(indatadir, [yourgridfilev]),)
# ------ Specify if there is a ice mask for high latitudes
#        (if true, mask is recomputed at each cycle)
ice_mask = False or True
```

(continues on next page)

(continued from previous page)

```
# ------ Type of grid:
#        'regular' or 'irregular', if 'regular' only 1d coordinates
#        are extracted from model
grid = 'regular'
# ------ Specify list of variable:
list_input_var = {'ucur': [[u_var], [vel_ext], 0], 'vcur': [[v_var], [v_ext], 1],
                  'uuss': [[uuss_var], [uss_ext], 0], 'vuss': [[vuss_var], [uss_ext],␣
↪1],
                  'ice': [[ice_var], [ice_ext], 0], 'mssd': [[mssd_var], [msd_ext],␣
↪0],
                  'mssx': [[mssx_var], [mss_ext], 0], 'mssy':[[mssy_var], [mss_ext]],
                  'ssh': [[ssh_var], [ssh_ext], 0],
                  'uwnd': [[uwnd_var], [wnd_ext], 0], 'vwnd': [[vwnd_var], [wnd_ext],␣
↪1]}
# ------ Specify longitude variable:
lon = ('longitude', 'longitude')
# ------ Specify latitude variable:
lat = ('latitude', 'latitude')
# ------ Specify number of time in file:
dim_time = 24
# ------ Time step between two model outputs (in days):
timestep = 1/24.
# ------ Number of outputs to consider:
#        (timestep*nstep=total number of days)
nstep = 35*24
# ------ Not a number value:
model_nan = -32767.
```

```
# ----------------------#
# SKIM output files
# ----------------------#
# ------ Output file root name:
#        (Final file name is root_name_c[cycle].nc
file_output = os.path.join(outdatadir, config)
# ------ 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' or 'linear'
# ------ List of output variables:
list_output = ['ssh_obs', 'ur_true', 'ucur', 'vcur', 'uuss', 'vuss', 'instr',
               'radial_angle', 'vwnd', 'mssx', 'mssy', 'mssxy', 'uwb',
               'ssh_true', 'ssh', 'ice', 'mssd',
               'vindice', 'ur_obs', 'uwnd', 'sigma0']
```

```
# ----------------------#
# SKIM 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  or os.path.join(outdatadir, 'Random_coeff.nc')
# Compute instrumental nadir noise:
nadir = True
# ------ Number of random realisations for instrumental and geophysical error
```

(continues on next page)

(continued from previous page)

```
#        (recommended ncomp=2000), ncomp1d is used for 1D spectrum, and ncomp2d
#        is used for 2D spectrum (wet troposphere computation):
ncomp1d = 3000
ncomp2d = 2000
# ------- Instrument white noise error
instr = True or False
# ------- Choice of instrument configuration
instr_configuration = 'A' or 'B'
# ------- Attitude error
attitude = True or False
# ------- File which provide the AOCS error:
yaw_file = os.path.join(dir_setup, 'sample_req1.nc')
# -- Geophysical error
## ----------------------
# ------ Consider ice in sigma0 computation
ice = True or False
# ------ Rain error (True to compute it):
rain = True or False
# ------ Rain file containing scenarii (python file):
rain_file = os.path.join(dir_setup, [yourrainscenarii])
# ------ Threshold to flag data:
rain_threshold = 0.15
# ------- Wave bias
uwb = True or False
```

```
# ------------------------#
# L2C computation
# ------------------------#
# config name for L2c:
config_l2c = '[yourl2cconfig]'
# Length resolution to select neighbors (in km):
resol = 40
# Grid resolution for l2c (alongtrack, acrosstrack) grid (in km):
posting = 5
# Remove noisy data around nadir (in km):
ac_threshold = 20
# List of variables to be interpolated on the swath:
list_input_var_l2c = {'ucur': ['ucur', 'cur', 0], 'vcur': ['vcur', 'cur', 1]}
```

```
# ------------------------#
# L2D computation
# ------------------------#
# config name for L2d:
config_l2d = ''
# Length resolution to select neighbors (multiplication factor):
resol_spatial_l2d = 1
# Temporal resolution to select neighbors (multiplication factor):
resol_temporal_l2d = 1
# Grid resolution for l2d (lat, lon) grid (in degrees):
posting_l2d = (0.1, 0.1)
# Time domain: (start_time, end_time, dtime) in days:
time_domain = (5, 25, 1)
# Spatial domain (lon_min, lon_max, lat_min, lat_max):
spatial_domain = [0, 360, -90, 90]
# List of variables to be interpolated on the grid:
list_input_var_l2d = {'ucur': ['ucur', 'cur', 0], 'vcur': ['vcur', 'cur', 1]}
```

2.6 References:

The SKIM team Sea surface KInematics Multiscale monitoring, full proposal for ESA EE9, 2017, The SKIM team

CHAPTER

THREE

EVOLUTION OF THE SIMULATOR

3.1 Changelog for skimulator

3.1.1 3.0

2019/07/10 - Handle ice and ocean areas separately

- Add configuration B for TED analysis
- Implement new version of Wave Doppler parametrisation
- Improve estimation of radial Stokes and mss (separation of ocean and ice, use 12º beam for mss computation)
- Add band of validity for ussr
- Improve l2c and l2d computation by filtering out low wind areas

3.1.2 2.9

2019/06/14 - Read rain files if they are provided in gridded netcdf at the same time steps at the OGM files.

- Add wind interpolation in l2c / l2d computation
- Add atmospheric dsigma error
- Add azimuthal dsigma error
- Add oi of all the noises separately for l2c computation
- Update diagnostics codes with all the new implemented errors
- Add remaining attitude noise

3.1.3 2.8

2019/05/15 - Implement Wave doppler inversion

- Correct azimuthal noise for high latitude and coastal areas
- Adapt oi to avoid holes in error free reconstruction
- Add second configuration for instrumental error computation
- Assume the radial Stokes reconstruction will be improved by 2/3

3.1.4 2.7

2019/04/31 - Add fit from TAS instrumental noise

- Implement wave doppler parametrisation from learning algorithms
- Add json parameter file for diagnostics routine
- Fix import issues in python 3.5, numpy.random handling for mac os
- Implement Fit TED

3.1.5 2.6

2019/03/08 - Adapt spatial and temporal filtering length with latitude for l2d computation

- Implement rain statistical scene for regional run
- Improve format (long name, units)
- Fix bugs in l2c (mask construction,)

3.1.6 2.5

2019/02/11 - L2c reconstruction parallelized

- Fix bugs in l2c computation (import issues, list of variables, mask l2c on coast using model)
- Add l2d computation from l2b input files, with offline interpolation function
- Fix for model data that are not correctly masked
- Add python scripts for l2b / l2d diagnostics

3.1.7 2.4

2019/01/03 - Fix bug when ice Flag is True

- Enable interpolation of variables with different coordinates
- Change listing of files: does not inculde the extension and suffix anymore

2018/12/31 - Refactoring to use the parallelization module in the grids generation step

- Isolate parallelisation code

2018/12/21 - Add option to avoid displaying progress bar while outputs are saved in a log file

- Raise error while using multiprocessing to make debugging easier
- Add ice floes consideration
- Save grid hash to detect if already processed grid are compliant with current parameter file

3.1.8 2.3

2018/12/14 - Add nadir-like observation with and without errors

- Add diagostic and plot example scripts (jupyter notebooks)

3.1.9 2.2

2018/12/07 - Change format of l2c

- Read satellite elevation and cycle from orbit files
- Add coherence diagnostics for L2c
- Fix typos

3.1.10 2.1

2018/10/17 - Handle any OGCM file withou waves (no noise is produced)

- Add rms diagnostics for L2c
- Add first time of model in parameter file to handle timestamps in netcdf files
- Add interpolation of model during regridding of L2c for diagnostic purposes
- Add true along track and across track velocity in L2c
- Handle nearly empty pass for L2c reconstruction
- Add error free plot for L2c std computation
- Add grid file in parameter file
- Add exponential window in space for L2c OI
- Fix bug (l2c sign issues on descending tracks, index time in l2c reconstruction)

3.1.11 2.0

2018/10/16 - Handle interpolation of an ensemble of variables and files provided in the parameter file

- Compute sigma and G from mss, wind, stockes . . . , compute instrumental noise from sigma0 and wave bias from stockes and G
- Compute remaining wave bias with simulating a correction using neighbors (errdcos function)
- Clean up by splitting codes and creation of new module mod_run
- Add module in test for diagnostics puposes (RMS for L2b and L2c)
- Add 2018 new configurations (2018_8a, b, c and 2018_6a)
- Correct satellite elevation used for Metop

- Fix bug in L2c during interpolation of descending tracks

1.31 — 2018/09/18 - Remove attenuation in sin(beam_angle) assuming that we will be able to correct it

- Fix bugs (typo in mod_tools, debug print, exceed dimension in time)

3.1.12 1.3

2018/07/18 - Add attenuation in sin(beam_angle) in projected radial velocity

- Improve packaging with a VERSION.txt that gather git information
- Add test script to generate L2C data

3.1.13 1.2

2018/07/01 - Handle generation of grid only when no model file is provided

- Implement error messages and user proof simulator
- Handle files with Arakawa grid C.
- Fix bugs (sorting index in multiprocessor to handle files in the right order, interpolation in global simulation, add errors to model, angle to compute instrumental rms)

3.1.14 1.1

2018/04/01 - Parallelisation

- Add figures in documentation and example files in example
- Fix bugs (issues with numpy 14.2, interpolation at 0E, radial velocity)

3.1.15 1.0

2017/12/20

- First public release
- Packaging improved
- Documentation

3.1.16 1.0 beta 1

2017/05/15 First beta version, undocumented

CHAPTER

FOUR

AUTO GENERATED DOCUMENTATION

4.1 Main program: run_simulator

Main program: Usage: run_simulator(file_param)

If no param file is specified, the default one is exemple/params_exemple.txt

In the first part of the program, model coordinates are read and the SKIM swath is computing accordingly.

The SKIM grid parameters are saved in netcdf files, if you don't want to recompute them, set maksgrid (in params file) to False.

In the second part of the program, errors are computed on SKIM grid for each pass, for each cycle. The error free velocity is the velocity interpolated from the model at each timestep. Note that there is no temporal interpolation between model files and thus if several files are used in the velocity interpolation, some discontinuities may be visible.

OUTPUTS are netcdf files containing the requested errors, the error free radial velocity and the radial velocity with errors. There is one file every pass and every cycle.

#——————————————————————————— # Additional Documentation # Authors: Lucile Gaultier # # Modification History: # - Mar 2017: Original by Lucile Gaultier, ODL # # Notes: # - written for Python 3.5, tested with Python 3.5, 3.7 # #——————————————————————————— Copyright (C) 2017-2018 OceanDataLab This file is part of skimulator.

skimulator is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

skimulator is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with skimulator. If not, see <http://www.gnu.org/licenses/>.

`skimulator.run_simulator.`**`err_formatter`**(*pid*, *grid*, *cycle*, *exc*)
Transform errors stored by the JobsManager into readable messages.

`skimulator.run_simulator.`**`exc_formatter`**(*exc*)
Format exception returned by sys.exc_info() as a string so that it can be serialized by pickle and stored in the JobsManager.

`skimulator.run_simulator.`**`make_skim_data`**(*_proc_count*, *jobs*, *die_on_error*, *progress_bar*)
Compute SWOT-like data for all grids and all cycle,

`skimulator.run_simulator.`**`run_simulator`**(*p*, *die_on_error=False*)
Main routine to run simulator, input is the imported parameter file, no outputs are returned but netcdf grid and data files are written as well as a skimulator.output file to store all used parameter.

4.2 Read data: rw_data

exception skimulator.rw_data.**IncompatibleGridError**(*path*, *grid_hash*, *params_hash*, **args*, ***kwargs*)

Raised

class skimulator.rw_data.**NETCDF_MODEL**(*p*, *ifile=None*, *list_input_var=None*, *lon=('longitude',)*, *lat=('latitude',)*, *depth=0*, *time=0*)

Class to read any netcdf data.

USAGE is NETCDF_MODEL(file=name of file ,var= variable name, lon=variable longitude, lat=variable latitude, units=).

Argument file is mandatory, arguments var, lon, lat are specified in params file.

calc_box(*p*)

Calculate subdomain coordinates from netcdf file Return minimum, maximum longitude and minimum, maximum latitude

read_coordinates(*p*, *index=None*)

Read coordinates from netcdf file

Argument is index=index to load part of the variable.

read_var(*p*, *ind_lon=None*, *index=None*)

Read variables from netcdf file

Argument is index=index to load part of the variable.

class skimulator.rw_data.**Sat_SKIM**(*ifile=None*, *lon=None*, *lat=None*, *lon_nadir=None*, *lat_nadir=None*, *time=None*, *cycle=None*, *al_cycle=None*, *x_al=None*, *timeshift=None*)

Sat_SKIM class: to read and write data that has been created by SKIM simulator

load_data(*p*, ***kwargs*)

Load swath variables stored in Satellite grid file sgridfile.

(longitude, latitude, number of days in a cycle, crossed distance during a cycle, time, along track and across track position).

load_swath(*p*, ***kwargs*)

Load swath variables stored in Satellite grid file sgridfile.

(longitude, latitude, number of days in a cycle, crossed distance during a cycle, time, along track and across track position).

write_data(*p*, *outdata*)

Write SKIM data in output file file_output Dimensions are x_al (along track distance), x_ac (across track distance).

Variables are longitude, latitude, index (file number), error-free radial velocity (velocity interpolated from the model and projected with the radial angle), selected errors (instrument, uss bias, radial uss) and velocity with errors.

write_swath(*p*, ***kwargs*)
Write swath location in Satellite grid file sgridfile.

Dimensions are time (i.e. along track), x_ac (across track) and cycle (1).

Variables are longitude, latitude, number of days in a cycle, distance crossed in a cycle, time, along track and across track distances are stored.

class skimulator.rw_data.**WW3**(*p*, *ifile=None*, *list_input_var=None*, *lon=('longitude',)*, *lat=('latitude',)*, *depth=0*, *time=0*)
Class to read ww3 netcdf data.

USAGE is NETCDF_MODEL(file=name of file ,var= variable name, lon=variable longitude, lat=variable latitude, units=).

Argument file is mandatory, arguments var, lon, lat are specified in params file.

calc_box(*p*)
Calculate subdomain coordinates from netcdf file Return minimum, maximum longitude and minimum, maximum latitude

read_coordinates(*p*, *index=None*)
Read coordinates from netcdf file

Argument is index=index to load part of the variable.

read_var(*p*, *ind_lon=None*, *index=None*)
Read variables from netcdf file

Argument is index=index to load part of the variable.

skimulator.rw_data.**read_coordinates**(*nfile*, *nlon*, *nlat*, *twoD=True*)
General routine to read coordinates in a netcdf file.

Inputs are file name, longitude name, latitude name.

Outputs are longitudes and latitudes (2D arrays).

skimulator.rw_data.**read_var**(*nfile*, *var*, *index=None*, *time=0*, *depth=0*, *model_nan=None*)
General routine to read variables in a netcdf file.

Inputs are file name, variable name, index=index to read part of the variables, time=time to read a specific time, depth=depth to read a specific depth, model_nan=nan value

skimulator.rw_data.**write_params**(*params*, *pfile*)
Write parameters that have been selected to run swot_simulator.

4.3 Build swath: build_swath

You should have received a copy of the GNU General Public License along with skimulator. If not, see <http://www.gnu.org/licenses/>.

skimulator.build_swath.**err_formatter**(*pid*, *ipass*, *cycle*, *exc*)
Transform errors stored by the JobsManager into readable messages.

skimulator.build_swath.**exc_formatter**(*exc*)
Format exception returned by sys.exc_info() as a string so that it can be serialized by pickle and stored in the JobsManager.

skimulator.build_swath.**make_skim_grid**(*_proc_count*, *jobs*, *die_on_error*, *progress_bar*)
Compute SWOT grids for every pass in the domain

skimulator.build_swath.**makeorbit**(*modelbox*, *p*, *orbitfile='orbit_292.txt'*, *filealtimeter=None*)
Computes the swath of SKIM satellites on a subdomain. The path of the satellite is given by the orbit file and the subdomain corresponds to the one in the model. Note that a subdomain can be manually added in the parameters file.

Inputs are satellite orbit (p.filesat), subdomain (modelbox), List of postion of six degree beams, list of position of twelve degree beams, rotation speed.

Outputs are netcdf files containing SKIM position (lon, lat number of days in a cycle, distance crossed in a cycle, time)

skimulator.build_swath.**orbit2swath**(*modelbox*, *p*, *orb*, *die_on_error*)
Computes the swath of SKIM satellites on a subdomain from an orbit. The path of the satellite is given by the orbit file and the subdomain corresponds to the one in the model. Note that a subdomain can be manually added in the parameters file.

Inputs are satellite orbit (p.filesat), subdomain (modelbox), Swath parameters (half gap distance p.halfgap, half swath distance p.halfswath, along track resolution p.delta_al, across track resolution p.delta_ac).

Outputs are netcdf files containing SKIM grid (along track distance x_al, radial angle, longitude lon and latitude lat, number of days in a cycle cycle, distance crossed in a cycle cycle_al, time

4.4 Build errors: build_error

Copyright (C) 2017-2018 OceanDataLab This file is part of skimulator.

skimulator is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

skimulator is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with skimulator. If not, see <http://www.gnu.org/licenses/>.

skimulator.build_error.**compute_wd_ai_par**(*output_var_i*, *radial_angle*, *beam_angle*)
Compute wave doppler using coefficients learned from ww3 data

skimulator.build_error.**compute_wd_ai_par_old**(*output_var_i*, *radial_angle*, *beam_angle*)
Compute wave doppler using coefficients learned from ww3 data

skimulator.build_error.**compute_wd_old_par**(*output_var_i*, *radial_angle*, *beam_angle*)
Compute Wave doppler using old parametrisation uwd = Gr * ussr and Gr = a * log(b + c/nwr) * (1- tanh(angle))

class skimulator.build_error.**error**(*p*, *instr=None*, *uss=None*, *err_uss=None*)
Class error define all the possible errors that can be computed using SKIM simulator. Random realisation of

errors can be initialized using init_error. If the random realisations have already been computed and stored in file file_coeff, the random realisations are read directly using load_coeff. The corresponding errors on a swath can be computed using make_error.

init_error(*p*)
Initialization of errors: Random realisation of errors are computed using a known power spectrum. The outputs are the amplitude, the phase and the frequency of each random realisation. By default, there are ncomp1d=2000 random realisations for the instrumental errors (1d spectrum) and ncomp2d=2000 random realisations for the geophysical errors (2d spectrum) and nrandkarin*x_ac km of random number for KaRIN noise.

make_error(*u_true*, *p*, *radial_angle*, *Gvar*, *file_rms_instr*, *uss=(None*, *None)*, *std_local=None*, *errd-cos=None*)
Build errors corresponding to each selected noise among the effect of the wet_tropo, the phase between the two signals, the timing error, the roll of the satellite, the sea surface bias, the distorsion of the mast, the karin noise due to the sensor itself.

make_vel_error(*ur_true*, *p*)
Compute observed velocity adding all the computed error to the model velocity.

class skimulator.build_error.**errornadir**(*p*, *nadir=None*, *wet_tropo1=None*, *wt=None*)
Class errornadir defines the error on the nadir. Random realisation of errors can be initialized using init_error. The correspondg errors on a swath can be computed using make_error.

init_error(*p*)
Initialization of errors: Random realisation of errors are computed using a known power spectrum. The outputs are the amplitude, the phase and the frequency of each random realisation. By default, there are ncomp2d=2000 random realisations for the wet tropo and ncomp1d=2000 random realisations for the nadir 1d spectrum error.

load_coeff(*p*)
Load existing random realisations that has been stored in nadir+file_coeff. The outputs are the amplitude, the phase and the frequency of each random realisation. There are ncomp random realisations.

save_coeff(*p*)
Save random realisations to enable runs to be reproducible. The ncomp1d random phase phi, amplitude A and frequency fr for 1D spectrum and ncomp2d random phase phi, amplitude A and frequencies frx and fry for 2D spectrum are saved in nadirfile_coeff for each error and can be loaded using load_coeff.

skimulator.build_error.**make_vel_error**(*ur_true*, *p*, *instr=None*, *err_uss=None*)
Compute observed velocity adding all the computed error to the model velocity.

4.5 Useful tools: mod_tools

skimulator.mod_tools.**cart2spher**(*x*, *y*, *z*)
Convert cartesian coordinates to spherical coordinates.

Inputs are cartiesian coordinates x, y, z.

Return lon, lat.

skimulator.mod_tools.**cart2sphervect**(*x*, *y*, *z*)
Convert cartesian coordinates to spherical coordinates.

Inputs are cartiesian coordinates x, y, z.

Return lon, lat.

skimulator.mod_tools.**gen_coeff_signal1d**(*f*, *PS*, *nc*)
Generate nc random coefficient from a spectrum PS with frequencies f.

Return Amplitude, phase and frequency of nc realisations

skimulator.mod_tools.**gen_coeff_signal2d**(*f*, *PS*, *nc*)
Generate nc random coefficient from a spectrum PS with frequencies f.

Inputs are: frequency [f], spectrum [PS], number of realisation [nc] Return Amplitude, phase and frequency in 2D (frx, fry) of nc realisations

skimulator.mod_tools.**load_python_file**(*file_path*)
Load a file and parse it as a Python module.

skimulator.mod_tools.**rotationmat3D**(*theta*, *axis*)
Creates a rotation matrix: Slow method.

Inputs are rotation angle theta and rotation axis axis. The rotation matrix correspond to a rotation of angle theta with respect to axis axis.

Return the rotation matrix.

skimulator.mod_tools.**spher2cart**(*lon*, *lat*)
Convert spherical coordinates to cartesian coordinates.

Inputs are longitude, latitude.

Return x, y, z

skimulator.mod_tools.**update_progress**(*progress*, *arg1*, *arg2*)
Creation of a progress bar: print on screen the progress of the run

skimulator.mod_tools.**update_progress_multiproc**(*status*, *info*)
Creation of progress bar: print on screen progress of run, optimized for parrallelised tasks

CHAPTER

FIVE

OPEN SOURCE LICENCE

This is the open source licence regarding the Open Source SKIM Simulator.

5.1 Software licence and copyright

GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <https://fsf.org/> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for

software and other kinds of works.

The licenses for most software and other practical works are designed

to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not

price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you

these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether

gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps:

(1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains

that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run

modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents.

States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and

modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of

works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this

License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work

in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based

on the Program.

To "propagate" a work means to do anything with it that, without

permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other

parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices"

to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work

for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official

standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other

than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all

the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users

can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that

same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of

copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not

convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under

the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological

measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid

circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the

work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you

receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey,

and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to

produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

> a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
>
> b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
>
> c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
>
> d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent

works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms

of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

> a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
>
> b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software

in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded

from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any

tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods,

procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or

specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a

requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided,

in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this

License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option

remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you

add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

> a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
>
> b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
>
> c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
>
> d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
>
> e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
>
> f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further

restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you

must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the

form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly

provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your

license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is

reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the

licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or

run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically

receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an

organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the

rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this

License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims

owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this

definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free

patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express

agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license,

and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or

arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within

the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting

any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others’ Freedom.

If conditions are imposed on you (whether by court order, agreement or

otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have

permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue

to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of

the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the

Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future

versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different

permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY

APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING

WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided

above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest

possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest

to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program. If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short

notice like this when it starts in an interactive mode:

> <program> Copyright (C) <year> <name of author> This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’. This is free software, and you are welcome to redistribute it under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school,

if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <https://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program

into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <https://www.gnu.org/licenses/why-not-lgpl.html>.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

S

U

W