The Arctic Ocean Observation Operator - ARC3O How to use it

Clara Burgard

March 16, 2020

1 Introduction

The Arctic Ocean Observation Operator (ARC3O) computes brightness temperatures at 6.9 GHz, vertical polarization, based on climate model output. More information about the motivation, structure and evaluation can be found in Burgard et al. (2020a) and Burgard et al. (2020b). Currently (October 2019), it is customized for MPI-ESM output but can be used for other models if the variable names are changed accordingly in the ARC3O functions.

It contains the following files:

- mask_functions.py
- profile_functions.py
- memls_functions_2D.py
- operation_satsim_functions.py

2 What python packages do I need?

To run ARC3O, you need to install the following python packages:

- numpy
- xarray
- pandas
- datetime
- time
- timeit

- itertools
- tqdm
- subprocess
- pathos.multiprocessing

3 How do I run it?

3.1 Prepare your data

Your MPI-ESM data should be divided into monthly files, found in one folder (inputpath). Also, to prepare the seasons and ice types mask, ARC3O needs one file in which all the data is merged.
These files should contain the following variables:

- snifrac: snow fraction on ice [0-1]
- siced: sea-ice thickness [m]
- sni: snow water equivalent [m]
- tsi: surface temperature at surface of snow (if present) or of ice (if no snow) [K]
- qvi: columnar water vapor [mm]
- wind10: wind speed [m/s]
- xlvi: columnar liquid water [mm]
- tsw: sea-surface temperature [K]
- seaice: sea-ice concentration [0-1]
- slm: sea-land mask
- ameltfrac: melt-pond fraction [0-1]

3.2 Running ARC3O

You can run ARC3O using different functions, depending on what length of period you want to compute your brightness temperatures. These functions can be found in "operation_satsim_functions.py". The function satsim_complete_parallel can be used in one line. It runs a loop over the years of interest (between start_year and end_year) and runs the 12 months in parallel. To do so it reads in the data divided in monthly files from inputpath:

INPUT:

- orig_data : MPI-ESM data whole period of interest (all years together)

- freq_of_int: 6.9 GHz. BE CAREFUL: ARC3O can run without error at another frequency of AMSR-E but the results and assumptions have not been checked at these frequencies and are given without a guarantee
- start_year: first year of interest in format yyyy
- end_year: last year of interest in format yyyy
- inputpath: where to find the monthly data
- outputpath: where to write out the results
- file_begin: what is the file string before "month+year" in the data file read in from inputpath
- file_end: what is the file string after "month+year" in the data file read in from inputpath
- timestep: what is the timestep of your data in hours (default is set to 6)
- write_mask: prepare the mask for ice type and seasons (default is yes). Set to "no" if you already computed this file in a previous iteration.
- write_profiles: compute temperature and salinity (etc.) profiles (default is yes). Set to "no" if you already computed these files in a previous iteration
- compute_memls: compute the brightness temperatures (default is yes).
- e_bias_fyi and e_bias_myi: bias correction factor for first-year and multiyear ice (default: e_bias_fyi=0.981,e_bias_myi=0.963)

OUTPUT (files in outputpath):

- period_masks_assim.nc: contains masks for ice type (open water, first-year ice, multiyear ice) and season (freezing, melting bare ice, melting snow)
- profiles_for_memls_snowyes_yyyymm.nc : profiles computed including the snow cover)
- profiles_for_memls_snowno_yyyymm.nc : profiles computed as if there was no snow cover)
- TB_assim_yyyymm_7.nc: contains brightness temperature at freq_of_int in V- and H-polarization (ignore H!) for the ice surfaces
- TBtot?_assim_yyyymm_7.nc: contains brightness temperature at freq_of_int in V- and H-polarization (ignore H!) at top of the atmosphere

If you only want to compute 1 month, you should use: satsim_complete_1month
start_year and end_year are not needed as input, instead you must give yy (year of interest in format yyyy) and mm (month of interest in format mm) as input

4 What exactly do the functions do?

4.1 mask_functions.py

- **ice_type_wholeArctic(sit,timestep)**
  This function returns an array representing the mask for ice types, using sea-ice thickness and timestep information. The ice is defined as multiyear ice, if there was ice in the last 365 days in that grid cell. It returns 1 for open water, 2 for first-year ice, and 3 for multiyear ice.

- **snow_period_masks(tsi,snow)**
  This function defines when there is snow growth and snow melt. It returns two arrays: one where the snow cover decreases and the second where the snow cover decreases AND the temperature is high enough for the snow melt (otherwise it could also be snow sublimation and snow ice formation).

- **summer_bareice_mask(sit,snow,timestep)**
  This function defines areas of bare ice in summer by looking for ice without a snow cover between April and September.

- **define_periods(sit,snow,tsi,timestep)**
  This function combines all season masking functions to have an overall season overview. The mask is as follows: 0 for open water, 1 for winter/freezing conditions, 2 for melting snow, 3 for bare ice in summer.

4.2 profile_functions.py

- **compute_ice_snow_int_temp(sit,snd,tsi)**
  This function computes the temperature at the snow-ice interface based on the snow and ice thickness and the temperature at the top of the snow. It is inspired from Semtner (1976).

- **build_temp_profile(surf_temp_ice,empty_temp_prof,winter_mask, fyi_mask,myi_mask,layer_amount,snow_opt,snd,sit,e_bias_fyi, e_bias_myi)**
  This function builds linear temperature profiles over depth between the snow surface temperature and ice surface temperature and between ice surface temperature and freezing temperature (-1.8°C).

- **build_thickness_profile(sit,snd,empty_thick_prof,winter_mask, layer_amount)**
  This function builds thickness profiles with equidistant layers over total ice thickness.

- **sal_approx_fy(norm_z)**
  This function builds a salinity profile as a function of depth for first-year ice. The formula is from Griewank and Notz (2015).

- **sal_approx_my(norm_z)**
  This function builds a salinity profile as a function of depth for multiyear ice. The formula is from Griewank and Notz (2015).

- **build_salinity_profile(empty_sal_prof,fyi_mask,myi_mask,winter_mask, layer_amount)**
  This function builds salinity profiles according to the ice type (given by the ice type masks) and formulas from Griewank and Notz (2015).

- **build_wetness_profile(empty_wet_prof,winter_mask)**
  This function builds wetness profiles (important for snow), currently everything is set to 0.

- **Sb(T)**
  This function computes the brine salinity from the tempature, based on formula from Notz (2005).

- **Vb(S,Sbr)**
  This function computes the liquid water fraction (=brine volume fraction), based on formula from Notz (2005).

- **icerho(T,S)**
  This function computes the sea-ice density, based on formula equations from Notz (2005).

- **build_density_profile(empty_dens_prof,temperature,salinity,winter_mask, layer_amount,snow_dens)**
  This function builds sea-ice and snow density profiles.

- **build_corrlen_profile(empty_corrlen_prof,prof_thick,fyi_mask,winter_mask, layer_amount)**
  This function builds correlation length profiles. Choice of values explained in Burgard et al. (2020a).

- **build_sisn_prof(empty_def_prof,myi_mask,fyi_mask,winter_mask, layer_amount)**
  This function builds layer type profiles. 1 is snow, 3 is first-year ice, 4 is multiyear ice

- **create_profiles(surf_temp_ice,sit,snow,layer_amount,info_ds,e_bias_fyi, e_bias_myi,snow_dens)**
  This function summarizes all profiles to write them out as an xr.Dataset. There is one set of profiles representing profiles for bare ice and one set of profiles representing profiles for snow-covered ice.

4.3 memls_functions_2D.py

These functions are mainly based on Wiesmann and Mätzler (1999), Ulaby et al. (1986) and Tonboe et al. (2006).

- **epsice(Ti,freq)**
  This function computes the dielectric permittivity of ice. After Hufford, Mitzima and Mätzler.

- **epsr(roi)**
  This function computes the real part of dielectric permittivity for dry snow from density.

- **ro2epsd(roi,Ti,freq)**
  This function computes the dielectric permittivity for dry snow from density.

- **mixmod(f,Ti,Wi,epsi,epsii)**
  This function computes the permittivity for Wetness > 0. Physical Mixing Model Weise 97 after Mätzler (1987) (corrected), water temperature is assumed constant at 273.15 K

- **epice(T,freq)**
  This function computes the dielectric constant of pure ice, after Mätzler (1998b).

- **Nsw(Ssw)**
  This function computes the normality of sea water or brine. Eq. 20 in Ulaby et al. (1986).

- **condbrine(T)**
  This function computes the conductivity of brine. Eq. 7 in Stogryn and Desargant (1985).

- **relaxt(T)**
  This function computes the relaxation time. Eq. 12 in Stogryn and Desargant (1985). The fit is valid up to -25°C.

- **epsib0(T)**
  This function computes the static dielectric constant of brine. Eq. 10 in Stogryn and Desargant (1985). The fit is valid up to -25°C.

- **ebrine(T,freq)**
  This function computes the brine permittivity. Eq. 1 in Stogryn and Desargant (1985). The fit is valid up to -25°C.

- **eice_s2p(e1,e2,v)**
  This function computes the effective dielectric constant of medium consisting of e1 and e2 improved born approximation by Mätzler (1998a). Polder/VanSanten mixing formulae for spherical inclusions.

- **sie(si,sal,Ti,freq,epsi,epsii)**
  This function computes the dielectric constant of ice if it is an ice layer. Background information in Ulaby et al. (1986).

- **mysie(si,rho,Ti,sal,freq,epsi,epsii)**
  This function computes the dielectric constant of ice if it is a multiyear ice layer. Background information in Ulaby et al. (1986).

- **abscoeff(epsi,epsii,Ti,freq)**
  This function computes the absorption coefficient from the dielectric properties. Background information in Ulaby et al. (1986).

- **tei_ndim(teta,ns)**
  This function has practical reasons, it appends teta at the end of ns over dimension layer_nb.

- **append_laydim_end(xrda,const)**
  This function has practical reasons, it appends a constant at the end of a dimension, here layer_nb.

- **append_laydim_begin(xrda,const)**
  This function has practical reasons, it appends a constant at the beginning of a dimension, here layer_nb.

- **pfadi(tei,di)**
  This function computes the effective path length in a layer.

- **fresnelc0(tei,epsi)**
  This function computes the fresnel reflection coefficients (assuming eps" = 0), layer n+1 is the air above the snowpack.

- **sccoeff(roi,Ti,pci,freq,Wi,gai,sccho)**
  This function computes the the scattering coefficient from structural parameters different algorithms can be chosen, by changing "sccho".

- **meteo_sc(si,rroi,rTi,rpci,freq,rWi,rgai,gbih,gbiv,gs6,ga2i)**
  This function computes the scattering coefficient of only partly recrystallized snow, linear combination of iborn, wahl==6.

- **iborn_s2p(e1,e2,eeff,v,k,pcc)**
  This function computes the improved born approximation by Mätzler (1998a), scattering coefficient of a collection of spherical inclusions articles with correlation length pcc using improved born approximation.

- **scice(si,gbih,gbiv,gs6,ga2i,Ti,sal,freq,pci)**
  This function computes the scattering coefficient from structural parameters.

- **scice_my(si,gbih,gbiv,gs6,ga2i,Ti,dens,freq,pci,sal)**
  This function computes the scattering coefficient of multiyear ice from structural parameters.

- **absorp2f(gbih,gbiv,gs6,ga2i,epsi,epsii,roi,Ti,pci,freq,Wi,gai)**
  This function computes the scattering coefficient from structural parameters

- **pfadc(teta,di,epsi,gs6)**
  This function computes the effective path length in a layer.

- **polmix(tscat,sih,siv)**
  This function computes the polarization mixing of the interface reflectivities of each layer (taking into account the first order scattering).

- **rt(gai,gbi,dei)**
  This function computes the layer reflectivity and transmissivity.

- **xr_diag(v,k=0)**
  This function is more technical, creates diagonal matrix with values of v on the diagonal, gives out an xr.DataArray.

- **build_xarray(data,temp)**
  This function is more technical, transforms np.array into xarray.

- **build_xarray_matrix2D(data,temp)**
  This function is more technical, transforms a np.array into xarray over the two dimensions layer_nb and matrix_dim for matrix operations.

- **xr_eye(v,k=0)**
  This function is more technical, creates diagonal matrix with ones on the diagonal, gives out an xr.DataArray.

- **xr_matmul(A,B,input_dims,output_dims)**
  This function is more technical, matrix multiplication for xarray, .dot gave weird results.

- **xr_linalg_inv(A)**
  This function is more technical, equivalent of np.linalg.inv for xarray.

- **add_matrix_dim(A,name_new_dim)**
  This function is more technical, adds a dummy matrix_dim to enable matrix multiplication.

- **layer(ri,s_i,ti,Ti,Tgnd,Tsky)**
  This function computes the upwelling brightness temperatures D (see Wiesmann and Mätzler (1999)).

- **memls_2D_1freq(freq,DI,TI,WI,ROI,PCI,SAL,SITYPE)**
  This function computes the brightness temperature and emissivity of a lon-lat-time-depth field at a given frequency.

4.4 operation_satsim_functions.py

- **prep_time(input_data)**
  This function transforms the date format given by MPI-ESM into a proper date format for xarray.

- **new_outputpath(log,outputpath,existing)**
  This function helps to organize different experiments (see example script).

- **prep_mask(input_data,write_mask,outputpath,timestep)**
  This function prepares or reads out the mask for the seasons and ice types.

- **prep_prof(input_data,write_profiles,info_ds,outputpath,yy,e_bias_fyi, e_bias_myi,snow_dens)**
  This function prepares or reads out the property profiles for use in MEMLS.

- **run_memls_2D(profiles,freq)**
  This function calls MEMLS and formats MEMLS results.

- **memls_module_general(yy,freq_of_int,profiles_yes,profiles_no,snifrac, barefrac,outputpath,compute_memls)**
  This function wraps around MEMLS to combine the brightness temperatures of bare and snow-covered ice.

- **compute_TBVice(info_ds,TB,pol)**
  This function combines the brightness temperature computed through MEMLS with other.brightness temperatures from other seasons ***!At the moment this is only valid for 6.9 GHz and V polarization!***

- **compute_emisV(info_ds,eice,pol)**
  This function combines the emissivity computed through MEMLS with other emissivities from other seasons ***!At the moment this is only valid for 6.9 GHz and V polarization!***

- **comp_F(m1,m2,W,pol)**
  This function is necessary for the atmospheric and ocean contribution. It computes the empirical term for any residual non-linear wind variations data for 6.9 GHz was not available so that we use the same values as for 10.7 GHz. Eq. 60 of Wentz and Meissner (2000).

- **amsr(V,W,L,Ta,Ts,TBV_ice,TBH_ice,e_icev,e_iceh,c_ice,freq,slm,mpf)**
  This function computes the atmospheric and ocean contribution. It is based on Wentz and Meissner (2000) and is tailored to AMSR2 frequencies. It was extended by C. Burgard to include melt ponds.

- **TB_tot(input_data,info_ds,memls_output,freq)**
  This function computes the brightness temperature at top of the atmosphere.

- **satsim_loop(file,yy,mm,info_ds,freq_of_int,e_bias_fyi,e_bias_myi, outputpath,write_profiles='no',compute_memls='yes',snow_dens=300.)**
  This function is where the actual brightness temperature simulation takes place.

- **compute_parallel(start_year,end_year,freq_of_int,e_bias_fyi,e_bias_myi, inputpath,outputpath,file_begin,file_end,info_ds,write_profiles, compute_memls)** This function parallelizes the brightness temperature simulation for the different months of a given year.

References

Burgard, C., D. Notz, L.T. Pedersen, and R.T. Tonboe (2020a). "The Arctic Ocean Observation Operator for 6.9 GHz (ARC3O) - Part 1: How to obtain sea-ice brightness temperatures at 6.9?GHz from climate model output". In: *The Cryosphere Discussions, in review*. DOI: 10.5194/tc-2019-317.

Burgard, C., D. Notz, L.T. Pedersen, and R.T. Tonboe (2020b). "The Arctic Ocean Observation Operator for 6.9 GHz (ARC3O) - Part 2: Development and evaluation". In: *The Cryosphere Discussions, in review*. DOI: 10.5194/tc-2019-318.

Griewank, P.J. and D. Notz (2015). "A 1-D modelling study of Arctic sea-ice salinity". In: *Cryosphere* 9, pp. 305–329. DOI: 10.5194/tc-9-305-2015.

Mätzler, C. (1987). "Applications of the interaction of microwaves with the natural snow cover". In: *Remote Sensing Reviews* 2, pp. 259–387. DOI: 10.1080/02757258709532086.

Mätzler, Christian (1998a). "Improved Born approximation for scattering of radiation in a granular medium". In: *Journal of Applied Physics* 83.11, pp. 6111–6117. DOI: 10.1063/1.367496.

Mätzler, Christian (1998b). "Microwave Properties of Ice and Snow". In: *Solar System Ices: Based on Reviews Presented at the International Symposium "Solar System Ices" held in Toulouse, France, on March 27–30, 1995*. Ed. by B. Schmitt, C. De Bergh, and M. Festou. Dordrecht: Springer Netherlands, pp. 241–257. ISBN: 978-94-011-5252-5. DOI: 10.1007/978-94-011-5252-5_10.

Notz, D. (2005). "Thermodynamic and Fluid-Dynamical Processes in Sea Ice". PhD thesis. University of Cambridge.

Semtner, Albert J. (1976). "A Model for the Thermodynamic Growth of Sea Ice in Numerical Investigations of Climate". In: *Journal of Physical Oceanography* 6.3, pp. 379–389. DOI: 10.1175/1520-0485(1976)006<0379:AMFTTG>2.0.CO;2.

Stogryn, A. and G. Desargant (1985). "The dielectric properties of brine in sea ice at microwave frequencies". In: *IEEE Transactions on Antennas and Propagation* 33 (5), pp. 523–532. DOI: 10.1109/TAP.1985.1143610.

Tonboe, R., S. Andersen, L. Toudal, and G. Heygster (2006). "Sea ice emission modelling". In: *Thermal Microwave Radiation - Applications for Remote Sensing*. Ed. by C. Mätzler, P.W. Rosenkranz, A. Battaglia, and J.P. Wigneron. IET Electromagnetic Waves Series 52, pp. 382–400.

Ulaby, F.T., R.K. Moore, and A.K. Fung (1986). "Passive microwave sensing of the ocean". In: *Microwave Remote Sensing, Active and Passive Volume III, From Theory to Applications*. Artech House, Inc. Chap. 18, pp. 1412–1521.

Wentz, F.J. and T. Meissner (2000). *Algorithm theoretical basis document (atbd), version 2*. Tech. rep. AMSR Ocean Algorithm, RSS Tech. Proposal 121599A-1. Remote Sensing Systems, Santa Rosa, CA.

Wiesmann, A. and C. Mätzler (1999). "Microwave emission model of layered snowpacks". In: *Remote Sens. Environ.* 70, pp. 307–316.