multimelt

Release 0.1

Clara Burgard

May 22, 2023

GETTING STARTED:

Multimelt contains a large amount of existing basal melt parameterisations for Antarctic ice shelves. The functions are written as such that the main input needed are temperature and salinity profiles in front of the ice shelf, the rest happens in the functions.

Also, multimelt contains functions to create masks of the Antarctic continent and the different ice shelves, and other geographical parameters on a circum-Antarctic scale, for input on a stereographic grid.

CHAPTER

ONE

DOCUMENTATION

1.1 About

1.1.1 The motivation behind multimelt

Ocean-induced melt at the base of Antarctic ice shelves (also called basal melt) is a crucial component for simulations of the Antarctic contribution to future sea-level evolution. Ice shelves have been thinning all around Antarctica in past decades. Thinning reduces the ice shelves' buttressing potential, which means that the restraining force that they exert on the ice outflow at the grounding line is lower and more ice is discharged into the ocean. In some bedrock configurations, increased melt can trigger instabilities. This is why it is currently the main uncertainty source in such projections.

To represent basal melt in numerical simulations, especially in ice-sheet models, parameterisations have been developed in past decades. They link a profile of temperature and salinity in front of an ice shelf to the melt rates below the ice shelf. The multimelt package provides code for the most commonly used of these parameterisations for rapid use and possible assessment or evaluation.

It takes input temperature and salinity profiles and returns several 2D and 1D metrics about the basal melt. It is designed to be applicable on circum-Antarctic scale.

1.1.2 Authors

The multimelt code is based on existing physical formulations:

- the linear fomulation as described by [Beckmann & Goosse, 2003]
- the quadratic formulation as described by [Holland et al., 2008], [DeConto & Pollard, 2016], [Favier et al., 2019] and [Jourdain et al., 2020]
- the plume parameterisation as described by [Lazeroms et al., 2018] and [Lazeroms et al., 2019]
- the box parameterisation (or PICO) as described by [Reese et al., 2018]
- the PICOP parameterisation as described py [Pelle et al., 2019]

An existing Fortran code by N. Jourdain has been translated and adapted to Python and further multimelt code has been developed by Clara Burgard - ClimateClara.

1.1.3 How to cite

The detailed description of the application of the functions in multimelt is found in [Burgard et al., 2022] and should therefore, when used, be cited as follows:

Burgard, C., Jourdain, N. C., Reese, R., Jenkins, A., and Mathiot, P. (2022): An assessment of basal melt parameterisations for Antarctic ice shelves, The Cryosphere, https://doi.org/10.5194/tc-16-4931-2022.

1.1.4 License

This project is licensed under the GPL3 License - see the license for details.

1.2 How to install

Currently, the easiest is to clone the multimelt git hub repository, go to the repository folder and type:

```
pip install .
```

If you want to modify it locally, you can fork the multimelt git hub repository, go to the repository folder and type:

```
pip install -e .
```

This package is programmed in python 3.8 and should be working with all *python versions > 3.8*. Additional requirements are numpy, xarray, pandas, tqdm and dask. (to be refined)

multimelt provides three types of "services". (1)It can produce masks of the different circum-Antarctic ice shelves and geometric properties on each ice shelf level (neede for the pdifferent parameterisations). (2) It computes mean profiles of temperature and salinity in front of the ice shelves in given domains of interest. (3) It computes 2D and 1D metrics related to basal melt of ice shelves.

The procedure to create the masks and box and plume characteristics is shown in the notebook `prepare_mask_example.ipynb`. The steps are also explained more in detail in *Preparing masks to identify ice shelf and main ice shelf characteristics around Antarctica*, *Preparing the box characteristics* and *Preparing the plume characteristics*.

The procedure to compute and format the input temperature (T) and salinity (S) profiles from 2D fields is shown in the notebook `T_S_profiles_per_ice_shelf.ipynb`. The steps are also explained more in detail in prepare_prof. !BE CAREFUL! If your 2D fields are in conservative temperature and absolute salinity, do not forget to convert them with the script `conversion_CTtoPT_SAtoSP.ipynb`.

The procedure to compute melt rates from temperature and salinity profiles is shown in the notebook `compute_melt_example.ipynb`. The steps are also explained more in detail in prod_melt.

1.3 How to run (1) : Preparing masks and geometric information

1.3.1 Preparing masks to identify ice shelf and main ice shelf characteristics around Antarctica

Input data

Currently, the mask functions of `multimelt` are tailored for **circum-Antarctic** model output from NEMO, the Nucleus for European Modelling of the Ocean [NEMO Team, 2019], interpolated to a south polar stereographic grid

(EPSG:3031). However, it can be used for other models/grids if the variable names are changed accordingly either in the functions or the model's output (requires a bit of digging).

The mask function needs these geometric variables as input:

- `file_msk`: file containing the circum-Antarctic information about basic masks with: 0 = ocean, 1 = ice shelves, 2 = grounded ice
- `file_bed_orig`: bathymetry [m], positive with depth
- `file_draft`: ice draft depth [m], positive with depth
- `file_isf_conc`: ice shelf concentration (if some grid cells are not fully covered with ice)
- `latlon_boundaries`: the latitude/longitude boundaries of the ice shelves, as defined for example in './mask_info/lonlat_masks.txt'
- `isf_metadata`: ice shelf name corresponding to ID in file above and data from about the different ice shelves, as shown in './mask_info/iceshelves_metadata_Nico.txt'
- `file_metadata_GL_flux`: flux across grounding line from , as shown in './mask_info/GL_flux_rignot13.csv'

Running

You can create an xr.Dataset containing the main geometric information with the function *multimelt.create_isf_mask_functions.create_mask_and_metadata_isf()* as follows:

```
import xarray as xr
import multimelt.create_isf_mask_functions as isfmf

inputpath_metadata = './multimelt/mask_info/'
outputpath_mask = # path where you want to store your mask netcdf file

file_bed_orig = # xr.DataArray containing the bathymetry (on grid EPSG:3031)
file_draft = # xr.DataArray containing the actual ice draft depth (not smoothed out␣
↪through a grid cell mean when the ice concentration is <1)
file_msk = # xr.DataArray containing mask: 0 = ocean, 1 = ice shelves, 2 = grounded ice␣
↪(on grid EPSG:3031)
file_isf_conc = # xr.DataArray containing the ice shelf concentration in each grid cell

xx = file_msk['x']
yy = file_msk['y']

whole_ds = isfmf.create_mask_and_metadata_isf(
                    file_msk, # file containing info about the grid (needs to be a␣
↪domain centered around the South Pole!)
                    -1*file_bed_orig, # negative bathymetry
                    file_msk, # original mask
                    -1*file_draft, # negative ice draft depth
                    file_isf_conc, # ice shelf concentration
                    False, # not chunked (CAREFUL! chunks not necessarily supported yet)
                    inputpath_metadata+'lonlat_masks.txt', # lon/lat boundaries of the␣
↪ice shelves
                    outputpath_mask, # output path for output to write out intermediate␣
↪steps
                    inputpath_metadata + 'iceshelves_metadata_Nico.txt', # file␣
↪containing name and Rignot data about the different ice shelves
```

(continues on next page)

(continued from previous page)

```
                    ground_point ='no', # if 'yes', the grounding line is defined on the␣
↪ice shelf points at the border to the ground
                    FRIS_one=True, # do you want to count Filchner-Ronne as one ice␣
↪shelf? True if yes, False if you want to have them as two separate ice shelves
                    variable_geometry=False, # TO BE USED FOR GEOMETRIES DIFFERENT FROM␣
↪PRESENT - if True, the ice shelves havee a slightly different geometry than present␣
↪and the limits have to be changed
                    write_ismask = 'yes', write_groundmask = 'yes', write_outfile='yes',
↪# if you already wrote one of these files, you can set option to 'no'
                    dist=40, # Defines the size of the starting square for the ground␣
↪mask - should be small if the resolution is coarse and high if the resolution is fine -
↪ can be modulated
                    add_fac=120, # Defines additional iterations for the propagation for␣
↪the ground mask - can be modulated
                    connectivity=4, # if variable_geometry = True:if 8 it looks at all 8␣
↪directions to see define neighbouring ice shelf points, if 4 only horizontally and␣
↪vertically
                    threshold=4, # if variable_geometry = True: an entity of 4 points is␣
↪considered as one ice shelf
                    write_metadata = 'yes' # writes out the file with only metadata
                    dist=40, # Defines the size of the starting square for the ground␣
↪mask - should be small if the resolution is coarse and high if the resolution is fine -
↪ can be modulated
                    add_fac=120 # Defines additional iterations for the propagation for␣
↪the ground mask - can be modulated
                                            )

# Write to netcdf
print('------- WRITE TO NETCDF -----------')
whole_ds.to_netcdf(outputpath_mask + 'mask_file.nc','w')
```

Output

The resulting netcdf file contains the following variables:

- `ISF_mask`: a map (on x and y) masking the ice shelves (0 for grounded, 1 for ocean, isf ID for ice shelves)
- `GL_mask`: a map (on x and y) masking the grounding line of the ice shelves (isf ID for grounding line, NaN elsewhere)
- `IF_mask`: a map (on x and y) masking the ice front of the ice shelves (isf ID for ice front, NaN elsewhere)
- `PP_mask`: a map (on x and y) masking the pinning points of the ice shelves (isf ID for pinning points, NaN elsewhere)
- `ground_mask`: a map (on x and y) masking mainland vs islands mask (0 for islands, 1 for ocean and ice shelves, 2 for mainland)
- `isf_name`: ice shelf name corresponding to ID in `ISF_mask`
- `isf_melt`: ice shelf melt as given in [] [Gt/yr]
- `melt_uncertainty`: ice shelf melt uncertainty as given in [] [Gt/yr]
- `isf_area_rignot`: ice shelf area as given in [] [km^2]

- `isf_area_here`: ice shelf area inferred from the input data [km^2]
- `ratio_isf_areas`: ratio isf area here/Rignot
- `front_bot_depth_max`: maximum depth between ice shelf draft and ocean bottom at the ice-shelf front [m]
- `front_bot_depth_avg`: average depth between ice shelf draft and ocean bottom at the ice-shelf front [m]
- `front_ice_depth_min`: minimum distance between sea surface and ice shelf front depth [m]
- `front_ice_depth_avg`: average distance between sea surface and ice shelf front depth [m]
- `front_min_lat`: Minimum latitude of the ice shelf front
- `front_max_lat`: Maximum latitude of the ice shelf front
- `front_min_lon`: Minimum longitude of the ice shelf front
- `front_max_lon`: Maximum longitude of the ice shelf front
- `dGL`: Shortest distance to respective grounding line [m]
- `dIF`: Shortest distance to respective ice front [m]
- `dGL_dIF`: Shortest distance to respective ice shelf front (only for grounding line points)

1.3.2 Preparing the box characteristics

Input data

The box and plume characteristics are inferred from the mask file `'mask_file.nc'` produced using *multimelt.create_isf_mask_functions.create_mask_and_metadata_isf()*.

```
import xarray as xr

whole_ds = xr.open_dataset(outputpath_mask + 'mask_file.nc')
```

In the NEMO case, we decide to focus on the ice shelves that are resolved enough on our grid, here the ones larger than 2500 km^2:

```
nonnan_Nisf = whole_ds['Nisf'].where(np.isfinite(whole_ds['front_bot_depth_max']),␣
↪drop=True).astype(int)
file_isf_nonnan = whole_ds.sel(Nisf=nonnan_Nisf)
large_isf = file_isf_nonnan['Nisf'].where(file_isf_nonnan['isf_area_here'] >= 2500,␣
↪drop=True) # only look at ice shelves with area larger than 2500 km2
file_isf = file_isf_nonnan.sel(Nisf=large_isf)
```

Running

```
import xarray as xr
import multimelt.box_functions as bf

outputpath_boxes = # path where you want to store your box characteristics netcdf file

file_draft = # xr.DataArray containing the actual ice draft depth (not smoothed out␣
↪through a grid cell mean when the ice concentration is <1)
file_isf_conc = # xr.DataArray containing the ice shelf concentration in each grid cell
```

(continues on next page)

(continued from previous page)

```
isf_var_of_int = file_isf[['ISF_mask', 'GL_mask', 'dGL', 'dIF', 'latitude', 'longitude',
↪'isf_name']]
out_2D, out_1D = bf.box_charac_file(file_isf['Nisf'], # ice shelf ID list
                                    isf_var_of_int, # variables of interest from file_isf
                                    -1*file_draft, # negative ice draft depth
                                    file_isf_conc, # ice shelf concentration
                                    outputpath_boxes, # output path for netcdfs
                                    max_nb_box=10 # maximum amount of boxes to explore
                                    )

print('------ WRITE TO NETCDF -------')
out_2D.to_netcdf(outputpath_boxes + 'boxes_2D.nc')
out_1D.to_netcdf(outputpath_boxes + 'boxes_1D.nc')
```

Output

The resulting netcdf file `boxes_2D.nc` contains the following variables:

- `dGL`: map (on x and y) of shortest distance to respective grounding line [m]
- `dIF`: map (on x and y) of shortest distance to respective ice front [m]
- `box_location`: map (on x and y) masking the location of box 1 to n, depending on the amount of boxes

The resulting netcdf file `boxes_1D.nc` contains the following variables:

- `box_area`: area of the respective box [m^2]
- `box_depth_below_surface`: mean depth at the top of the box [m]
- `nD_config`: amount of boxes that can be used in the config levels, according to the criteria that all boxes should have an area of more than 0 and that the box depth below surface has an ascending slope from grounding line to ice front.

1.3.3 Preparing the plume characteristics

Input data

The box and plume characteristics are inferred from the mask file `'mask_file.nc'` produced using *multimelt.create_isf_mask_functions.create_mask_and_metadata_isf()*.

```
import xarray as xr

whole_ds = xr.open_dataset(outputpath_mask + 'mask_file.nc')
```

In the NEMO case, we decide to focus on the ice shelves that are resolved enough on our grid, here the ones larger than 2500 km^2:

```
nonnan_Nisf = whole_ds['Nisf'].where(np.isfinite(whole_ds['front_bot_depth_max']),␣
↪drop=True).astype(int)
file_isf_nonnan = whole_ds.sel(Nisf=nonnan_Nisf)
```

(continues on next page)

(continued from previous page)

```
large_isf = file_isf_nonnan['Nisf'].where(file_isf_nonnan['isf_area_here'] >= 2500,␣
↪drop=True) # only look at ice shelves with area larger than 2500 km2
file_isf = file_isf_nonnan.sel(Nisf=large_isf)
```

Running

```
import xarray as xr
import multimelt.plume_functions as pf

plume_param_options = ['cavity','lazero', 'local']
# 'cavity': deepest grounding line, cavity slope
# 'lazero': grounding line and slope inferred according to Lazeroms et al., 2018
# 'local': grounding line inferred according to Lazeroms et al., 2018 and local slope

plume_var_of_int = file_isf[['ISF_mask', 'GL_mask', 'IF_mask', 'dIF', 'dGL_dIF',
↪'latitude', 'longitude', 'front_ice_depth_avg']]

# Compute the ice draft
file_draft = # xr.DataArray containing the actual ice draft depth (not smoothed out␣
↪through a grid cell mean when the ice concentration is <1)
ice_draft_pos = file_draft
ice_draft_neg = -1*ice_draft_pos

plume_charac = pf.prepare_plume_charac(plume_param_options,
                                       ice_draft_pos,
                                       plume_var_of_int
                                       )

print('------ WRITE TO NETCDF -------')
plume_charac.to_netcdf(outputpath_plumes+'plume_characteristics.nc')
```

Output

The resulting netcdf file `plume_characteristics.nc` contains the following variables:

- `zGL`: map (on x and y) of grounding line depth (negative downwards) associated to each ice shelf point [m]
- `alpha`: map (on x and y) of slope associated to each ice shelf point

multimelt provides three types of "services". (1)It can produce masks of the different circum-Antarctic ice shelves and geometric properties on each ice shelf level (neede for the pdifferent parameterisations). (2) It computes mean profiles of temperature and salinity in front of the ice shelves in given domains of interest. (3) It computes 2D and 1D metrics related to basal melt of ice shelves.

The procedure to create the masks and box and plume characteristics is shown in the notebook `prepare_mask_example.ipynb`. The steps are also explained more in detail in *Preparing masks to identify ice shelf and main ice shelf characteristics around Antarctica*, *Preparing the box characteristics* and *Preparing the plume characteristics*.

The procedure to compute and format the input temperature (T) and salinity (S) profiles from 2D fields is shown in the notebook `T_S_profiles_per_ice_shelf.ipynb`. The steps are also explained more in detail in prepare_prof. !BE CAREFUL! If your 2D fields are in conservative temperature and absolute salinity, do not forget to convert them with the script `conversion_CTtoPT_SAtoSP.ipynb`.

The procedure to compute melt rates from temperature and salinity profiles is shown in the notebook `compute_melt_example.ipynb`. The steps are also explained more in detail in prod_melt.

1.4 How to run (2) : Preparing the input profiles

1.4.1 Conversion from conservative temperature to potential temperature and from absolute salinity to practical salinity

Input data

Currently, most of the functions of `multimelt` are tailored for **circum-Antarctic** model output from NEMO, the Nucleus for European Modelling of the Ocean [NEMO Team, 2019], interpolated to a south polar stereographic grid (EPSG:3031). However, it can be used for other models/grids if the variable names are changed accordingly either in the functions or the model's output (requires a bit of digging).

The conversion script `conversion_CTtoPT_SAtoSP.ipynb` needs these files as input:

- `file_mask`: file containing the depth information of your depth coordinate, for each x/y pair if it is not constant (we will assume that it is approximately constant and take a mean over all x/y pairs at each depth level)
- `file_isf`: file containing masks created earlier with *multimelt.create_isf_mask_functions.create_mask_and_metadata_isf()*
- `file_TS_orig`: 3D field of conservative temperature, absolute salinity and sea-surface temperature for just one time step
- `ts_files`: 3D fields of conservative temperature and absolute salinity for all time steps (can be several files), they need to contain the 3D potential temperature `votemper`, the 3D practical salinity `vosaline`, and the 2D sea-surface temperature `sosst`

Running

Use the notebook `conversion_CTtoPT_SAtoSP.ipynb`.

Output

The resulting netcdf file contains the following variables:

- `theta_ocean`: 3D field of potential temperature
- `salinity_ocean`: 3D field of practical salinity

1.4.2 Prepare input temperature and salinity profiles averaged over given domains in front of the ice shelves

Input data

Currently, most of the functions of `multimelt` are tailored for **circum-Antarctic** model output from NEMO, the Nucleus for European Modelling of the Ocean [NEMO Team, 2019], interpolated to a south polar stereographic grid (EPSG:3031). However, it can be used for other models/grids if the variable names are changed accordingly either in the functions or the model's output (requires a bit of digging).

The input profiles are currently computed averaged over five domains: within 10, 25, 50 and 100 km of the ice-shelf front on the continental shelf, and offshore (for bathymetry deeper than the continental shelf).

The computation script `T_S_profiles_per_ice_shelf.ipynb` (1) calculates the distance to the ice front for the small domain in front of the ice shelf and (2) takes the ocean points at a given distance of the ice front and averages over them.It needs these files as input:

- `file_mask_orig`: file containing the variable `bathy_metry`
- `file_isf_orig`: file containing masks created earlier with *multimelt.create_isf_mask_functions.create_mask_and_metadata_isf()*
- `T_S_ocean_oneyear` and `T_S_ocean_files`: 3D field of potential temperature `theta_ocean` and practical salinity `salinity_ocean`

Running

Use the notebook `T_S_profiles_per_ice_shelf.ipynb`.

Output

The main result is the netcdf `'T_S_mean_prof_corrected_km_contshelf_and_offshore_1980-2018.nc'` containing profiles of `theta_ocean` and `salinity_ocean` averaged over different domains in front of the ice shelf (currently: within 10, 25, 50 and 100 km of the ice-shelf front on the continental shelf, and offshore for bathymetry deeper than the continental shelf).

It produces several temporary netcdf files needed in case the script crashes but that can be deleted afterwards:

- `'dist_to_ice_front_only_contshelf_oneFRIS.nc'`
- `'mask_offshore_oneFRIS.nc'`
- `'ds_sum_for_mean_contshelf.nc'`
- `'T_S_mean_prof_corrected_km_contshelf_1980-2018.nc'`
- `'ds_sum_for_mean_offshore.nc'`
- `'T_S_mean_prof_corrected_km_offshore_1980-2018.nc'`

multimelt provides three types of "services". (1)It can produce masks of the different circum-Antarctic ice shelves and geometric properties on each ice shelf level (neede for the pdifferent parameterisations). (2) It computes mean profiles of temperature and salinity in front of the ice shelves in given domains of interest. (3) It computes 2D and 1D metrics related to basal melt of ice shelves.

The procedure to create the masks and box and plume characteristics is shown in the notebook `prepare_mask_example.ipynb`. The steps are also explained more in detail in *Preparing masks to identify ice shelf and main ice shelf characteristics around Antarctica*, *Preparing the box characteristics* and *Preparing the plume characteristics*.

The procedure to compute and format the input temperature (T) and salinity (S) profiles from 2D fields is shown in the notebook `T_S_profiles_per_ice_shelf.ipynb`. The steps are also explained more in detail in prepare_prof. !BE CAREFUL! If your 2D fields are in conservative temperature and absolute salinity, do not forget to convert them with the script `conversion_CTtoPT_SAtoSP.ipynb`.

The procedure to compute melt rates from temperature and salinity profiles is shown in the notebook `compute_melt_example.ipynb`. The steps are also explained more in detail in prod_melt.

1.5 How to run (3) : Computing basal melt rates

The melt function is designed to receive information about the ice-shelf geometry (formatted with *multimelt.create_isf_mask_functions.create_mask_and_metadata_isf()*, *multimelt.plume_functions.prepare_plume_charac()*, and *multimelt.box_functions.box_charac_file()*), and about temperature and salinity profiles in front of one or several ice shelves (formatted in `T_S_profiles_per_ice_shelf.ipynb`). The output is a range of 2D and 1D variables describing the melt rates at the base of the ice shelf or ice shelves, if you have several.

1.5.1 Input data

To compute the melt, in *multimelt.melt_functions.calculate_melt_rate_1D_and_2D_all_isf()*), you need

> the list of ice shelf IDs you are interested in (`nisf_list`)

> 2D geometric info (`geometry_info_2D` :

- The variables contained in `'plume_charac_file.nc'` created as shown in *Preparing the plume characteristics*
- The variables `'ISF_mask'`, `'latitude'` and `'longitude'` contained in `'mask_file.nc'` created as shown in *Preparing masks to identify ice shelf and main ice shelf characteristics around Antarctica*
- `ice_draft_pos`: ice draft depth (positive downwards)
- `grid_cell_area_weighted`: area of the grid cells weighted with the ice shelf concentration
- `isfdraft_conc`: ice shelf concentration

> a stacked mask of the ice shelf regions to reduce calculation time and only select ice shelf point (`isf_stack_mask`). Can be created as follows:

```
isf_stack_mask = multimelt.useful_functions.create_stacked_mask(ISF_mask, nisf_list, ['y
↪','x'], 'mask_coord')
```

> 1D geometric info (`geometry_info_1D`) :

- The variables `'front_bot_depth_avg'`, `'front_bot_depth_max'`, and `'isf_name'` contained in `'mask_file.nc'` created as shown in *Preparing masks to identify ice shelf and main ice shelf characteristics around Antarctica*

> Temperature and salinity profiles (`T_S_profile`): one dataset containing `theta_ocean`, the potential temperature in °C, and `salinity_ocean`, the practical salinity in psu, both over at least the dimensions `Nisf` and `depth`.

> Input parameters to the different parameterisations: e.g. `gamma`, `E0`, `C`

If you want to use the box or PICOP parameterisation

> *The variables contained in* `'box_charac_file.nc'` *created as shown in Preparing the box characteristics*

1.5.2 Running

To run the simple parameterisations, use the following command

```
nisf_list = geometry_info_1D.Nisf
T_S_profile = file_TS.ffill(dim='depth')

mparam = # POSSIBILITIES: ['linear_local', 'quadratic_local', 'quadratic_local_locslope',
↪'quadratic_local_cavslope', 'quadratic_mixed_mean', 'quadratic_mixed_locslope','quadratic_
↪mixed_cavslope']

gamma = # fill in
ds_2D, ds_1D = meltf.calculate_melt_rate_1D_and_2D_all_isf(nisf_list,
                                                           T_S_profile, g
                                                           geometry_info_2D,
                                                           geometry_info_1D,
                                                           isf_stack_mask,
                                                           mparam,
                                                           gamma,
                                                           U_param=True)

ds_2D.to_netcdf(outputpath_melt+'melt_rates_2D_'+mparam+'.nc')
ds_1D.to_netcdf(outputpath_melt+'melt_rates_1D_'+mparam+'.nc')
```

To run the plume parameterisations, use the following command

```
nisf_list = geometry_info_1D.Nisf
T_S_profile = file_TS.ffill(dim='depth')

mparam = # POSSIBILITIES: ['lazero19_2', 'lazero19_modif2']

gamma = # fill in
E0 = # fill in

ds_2D, ds_1D = meltf.calculate_melt_rate_1D_and_2D_all_isf(nisf_list,
                                                           T_S_profile,
                                                           geometry_info_2D,
                                                           geometry_info_1D,
                                                           isf_stack_mask,
                                                           mparam,
                                                           gamma,
                                                           E0=E0,
                                                           verbose=True)

ds_2D.to_netcdf(outputpath_melt+'melt_rates_2D_'+mparam+'.nc')
ds_1D.to_netcdf(outputpath_melt+'melt_rates_1D_'+mparam+'.nc')
```

To run the box parameterisations, use the following command

```
nisf_list = geometry_info_1D.Nisf
T_S_profile = file_TS.ffill(dim='depth')
picop_opt = 'no'

nD_config = # POSSIBILITIES: 1 to 4
pism_version = # POSSIBILITIES: 'yes' or 'no'

mparam = 'boxes_'+str(nD_config)+'_pism'+pism_version+'_picop'+picop_opt

C = # fill in
gamma = # fill in

ds_2D, ds_1D = meltf.calculate_melt_rate_1D_and_2D_all_isf(nisf_list,
                                                           T_S_profile,
                                                           geometry_info_2D,
                                                           geometry_info_1D,
                                                           isf_stack_mask,
                                                           mparam,
                                                           gamma,
                                                           C=C,
                                                           angle_option='appenB',
                                                           box_charac_2D=box_charac_all_
↪2D,
                                                           box_charac_1D=box_charac_all_
↪1D,
                                                           box_tot=nD_config,
                                                           box_tot_option='nD_config',
                                                           pism_version=pism_version,
                                                           picop_opt=picop_opt)

ds_2D.to_netcdf(outputpath_melt+'melt_rates_2D_'+mparam+'.nc')
ds_1D.to_netcdf(outputpath_melt+'melt_rates_1D_'+mparam+'.nc')
```

To run the PICOP parameterisations, use the following command

```
nisf_list = geometry_info_1D.Nisf
T_S_profile = file_TS.ffill(dim='depth')

nD_config = # POSSIBILITIES: 1 to 4
pism_version = # POSSIBILITIES: 'yes' or 'no'
picop_opt = # POSSIBILITIES: '2018' or '2019'

mparam = 'boxes_'+str(nD_config)+'_pism'+pism_version+'_picopyes'

C = # for box part - fill in
gamma = # for box part - fill in

gamma_plume = # for plume part - fill in
E0 = # for plume part - fill in

ds_2D, ds_1D = meltf.calculate_melt_rate_1D_and_2D_all_isf(nisf_list,
                                                           T_S_profile,
                                                           geometry_info_2D,
```

(continues on next page)

(continued from previous page)

```
                                                        geometry_info_1D,
                                                        isf_stack_mask,
                                                        mparam,
                                                        gamma,
                                                        C=C,
                                                        E0=E0,
                                                        angle_option='appenB',
                                                        box_charac_2D=box_charac_all_
↪2D,
                                                        box_charac_1D=box_charac_all_
↪1D,
                                                        box_tot=nD_config,
                                                        box_tot_option='nD_config',
                                                        pism_version=pism_version,
                                                        picop_opt=picop_opt,
                                                        gamma_plume=gamma_plume)

ds_2D.to_netcdf(outputpath_melt+'melt_rates_2D_'+mparam+'.nc')
ds_1D.to_netcdf(outputpath_melt+'melt_rates_1D_'+mparam+'.nc')
```

1.5.3 Output

The xr.Dataset `ds_2D` contains the following variables on a map (2D):

- `melt_m_ice_per_s`: melt rate in m ice per second
- `melt_m_ice_per_y`: melt rate in m ice per year (computed per default but can also be removed by re-defining the list `options_2D` given to *multimelt.melt_functions.calculate_melt_rate_1D_and_2D_all_isf()*)
- `melt_m_we_per_y`: melt rate in m water equivalent per year (computed per default but can also be removed by re-defining the list `options_2D` given to *multimelt.melt_functions.calculate_melt_rate_1D_and_2D_all_isf()*)

The xr.Dataset `ds_1D` contains the following integrated variables (1D): * `melt_m_ice_per_y_tot`: total (accumulated) melt over each ice shelf in m ice per year * `melt_m_ice_per_y_avg`: average melt for each ice shelf in m ice per year (computed per default but can also be removed by re-defining the list `options_1D` given to *multimelt.melt_functions.calculate_melt_rate_1D_and_2D_all_isf()*) * `melt_m_ice_per_y_min`: minimum melt for each ice shelf in m ice per year (computed per default but can also be removed by re-defining the list `options_1D` given to *multimelt.melt_functions.calculate_melt_rate_1D_and_2D_all_isf()*) * `melt_m_ice_per_y_max`: maximum melt for each ice shelf in m ice per year (computed per default but can also be removed by re-defining the list `options_1D` given to *multimelt.melt_functions.calculate_melt_rate_1D_and_2D_all_isf()*) * `melt_we_per_y_tot`: total (accumulated) melt over each ice shelf in m water equivalent per year (computed per default but can also be removed by re-defining the list `options_1D` given to *multimelt.melt_functions.calculate_melt_rate_1D_and_2D_all_isf()*) * `melt_Gt_per_y_tot`: total melt over each ice shelf in Gt per year (computed per default but can also be removed by re-defining the list `options_1D` given to *multimelt.melt_functions.calculate_melt_rate_1D_and_2D_all_isf()*)

1.6 API References

1.6.1 Package main functions

1.6.2 T_S_profile_functions

1.6.3 box_functions

multimelt.box_functions.**box_charac_file**(*Nisf_list*, *isf_var_of_int*, *ice_draft*, *isf_conc*, *outputpath_boxes*, *max_nb_box=10*)

Compute box characteristics for all ice shelves of interest and prepare dataset to be written to netcdf (with all attributes).

This function writes two files containting the 1D and 2D relevant geometric information to compute the melt with the box model for all ice shelves of interest.

Parameters

- **Nisf_list** (*list of int*) – List of IDs of the ice shelves of interest
- **isf_var_of_int** (*xarray.Dataset*) – Dataset containing the variables GL_mask and ISF_mask with the coordinates ['x','y']
- **ice_draft** (*xarray.DataArray*) – Ice draft in m. The value must be negative under water.
- **isf_conc** (*floats between 0 and 1*) – Concentration of ice shelf in each grid cell
- **outputpath_boxes** (*str*) – Path to the fodler where the individual files should be written to. max_nb_box : int

  Amount of boxes in the configuration with the most boxes. Default is 10.

Returns

- **ds_2D** (*xarray.Dataset*) – Dataset containing the following variables for all ice shelves: dGL, dIF, box_location
- **ds_1D** (*xarray.Dataset*) – Dataset containing the following variables for all ice shelves:box_area, box_depth_below_surface, nD_config

multimelt.box_functions.**box_nb_like_reese**(*Nisf_list*, *dGL_all*, *dGL_max*, *n_max*, *file_isf*)

Compute the number of boxes for each ice shelf according to the criterion given in Eq. 9 in Reese et al. 2018.

Parameters

- **Nisf_list** (*array of int*) – List containing the ice shelf IDs for all ice shelves of interest.
- **dGL_all** (*array*) – Distance between each point and the closest grounding line.
- **dGL_max** (*float*) – Maximum distance between a point and the closest grounding line in the whole domain.
- **n_max** (*int*) – Maximum possible amount of boxes.
- **file_isf** (*xr.Dataset*) – mask_file containing ISF_mask.

Returns
nD_all – Corresponding amount of boxes for each ice shelf of Nisf_list.

Return type
xr.DataArray

multimelt.box_functions.**check_boxes**(*box_characteristics*, *upper_bnd*)

Check different box configurations.

This sets the number of boxes that can be applied to the ice shelf cavity depending on upper_bnd

Parameters

- **box_characteristics** (*xarray.Dataset*) – dataset containing 'box_area_tot' and 'box_depth_below_surface' for a given ice shelf
- **upper_bnd** (*int*) – maximum of box number wanted for that configuration

Returns

nD – closest number of boxes to upper_bnd possible given the box characteristics

Return type

int

multimelt.box_functions.**distance_isf_points_from_line**(*whole_domain*, *isf_points_da*, *line_points_da*)

Compute the distance between ice shelf points and a line.

This function computes the distance between ice shelf points and a line. This line can be the grounding line or the ice shelf front.

Parameters

- **whole_domain** (*xarray.DataArray*) – ice-shelf mask - all ice shelves are represented by a number, all other points (ocean, land) set to nan
- **isf_points_da** (*xarray.DataArray*) – array containing only points from one ice shelf
- **line_points_da** (*xarray.DataArray*) – mask representing the grounding line or ice shelf front mask corresponding to the ice shelf selected in isf_points_da

Returns

xr_dist_to_line – distance of the each ice shelf point to the given line of interest

Return type

xarray.DataArray

multimelt.box_functions.**loop_box_charac**(*Nisf_list*, *isf_var_of_int*, *ice_draft*, *isf_conc*, *outputpath_boxes*, *max_nb_box=10*)

Create xarray datasets with the box characteristics for all ice shelves of interest.

This function combines the box characteristics of all individual ice shelves into one dataset.

Parameters

- **Nisf_list** (*list of int*) – List of IDs of the ice shelves of interest
- **isf_var_of_int** (*xarray.Dataset*) – Dataset containing the variables ISF_mask with the coordinates ['x','y']
- **ice_draft** (*xarray.DataArray*) – Ice draft in m. The value must be negative under water.
- **isf_conc** (*floats between 0 and 1*) – Concentration of ice shelf in each grid cell
- **outputpath_boxes** (*str*) – Path to the fodler where the individual files should be written to. max_nb_box : int

  Amount of boxes in the configuration with the most boxes. Default is 10.

Returns

- **out_ds_2D** (*xarray.Dataset*) – Dataset containing the following variables for all ice shelves: `dGL`, `dIF`, `box_location`
- **out_ds_1D** (*xarray.Dataset*) – Dataset containing the following variables for all ice shelves:`box_area`, `box_depth_below_surface`, `nD_config`

multimelt.box_functions.**prepare_box_charac**(*kisf*, *isf_var_of_int*, *ice_draft*, *isf_conc*, *dx*, *dy*, *max_nb_box=10*)

Prepare file with the box characteristics for one ice shelf.

This function prepares the Dataset containing the relevant geometric information to compute the melt with the box model for an ice shelf of interest.

Parameters

- **kisf** (*int*) – ID of the ice shelf of interest
- **isf_var_of_int** (*xarray.Dataset*) – Dataset containing the variables `ISF_mask` with the coordinates `['x','y']` and `isf_name`
- **ice_draft** (*xarray.DataArray*) – Ice draft in m. The value must be negative under water.
- **isf_conc** (*floats between 0 and 1*) – Concentration of ice shelf in each grid cell
- **dx** (*float*) – Grid spacing in the x-direction
- **dy** (*float*) – Grid spacing in the y-direction max_nb_box : int

  Amount of boxes in the configuration with the most boxes

Returns

- **ds_2D** (*xarray.Dataset*) – Dataset containing the following variables for the ice shelf of interest: `dGL`, `dIF`, `box_location`
- **ds_1D** (*xarray.Dataset*) – Dataset containing the following variables for the ice shelf of interest:`box_area`, `box_depth_below_surface`, `nD_config`

1.6.4 constants

1.6.5 create_isf_mask_functions

This is a script to collect the masking steps with functions

@author: Clara Burgard

multimelt.create_isf_mask_functions.**combine_mask_metadata**(*df1*, *outfile*, *ds_time=False*)

Combine the metadata and the mask info into one netcdf.

This function combines the metadata and the mask info into one netcdf.

Parameters

- **df1** (*pandas.DataFrame*) – DataFrame containing 1D info for each ice shelf
- **outfile** (*xr.Dataset*) – Dataset containing all the produced masks: ISF_mask, GL_mask, IF_mask, PP_mask, ground_mask
- **ds_time** (*xr.Dataset*) – Dataset containing metadata for each time step in the case there is a time evolution of the geometry

Returns

whole_ds – Dataset containing all information from df1 and outfile

Return type
xr.Dataset

multimelt.create_isf_mask_functions.**compute_dist_front_bot_ice**(*mask_gline*, *mask_front*, *file_draft*, *file_bed*, *df1*, *lon*, *lat*, *dx=False*, *dy=False*, *new_mask=False*, *file_conc=False*)

Compute the depth of the bedrock and of the ice draft at the ice front.

This function computes the average and maximum depth of the bedrock and of the ice draft at the ice front.

Parameters

- **mask_gline** (*xr.DataArray*) – Array showing the grounding line with the ID of the corresponding ice shelf.
- **mask_front** (*xr.DataArray*) – Array showing the ice front with the ID of the corresponding ice shelf.
- **file_draft** (*xr.DataArray*) – Array containing the ice draft depth at at least at each ocean/ice shelf. Ice draft depth should be negative when below sea level!
- **file_bed** (*xr.DataArray*) – Array containing the bedrock topography at least at each ocean/ice shelf point. Bedrock depth should be negative when below sea level!
- **df1** (*pd.DataFrame*) – DataFrame containing the following columns for each ice shelf: columns=['isf_name', 'region', 'isf_melt', 'melt_uncertainty','isf_area_rignot']
- **lon** (*xr.DataArray*) – Longitude (depends on x,y for stereographic)
- **lat** (*xr.DataArray*) – Latitude (depends on x,y for stereographic)

Returns
df1 – DataFrame containing the following columns for each ice shelf: columns=['isf_name', 'region', 'isf_melt', 'melt_uncertainty','isf_area_rignot'] AND ['front_bot_depth_max','front_bot_depth_avg','front_ice_depth_min','front_ice_depth_avg','front_min_lat','front_max

Return type
pd.DataFrame

multimelt.create_isf_mask_functions.**compute_distance_GL_IF_ISF**(*whole_ds*)

Compute the distance from each point to the grounding line and the ice front.

This function computes the distance between each point and the grounding line on the one hand and the ice front on the other hand.

Parameters
whole_ds (*xr.Dataset*) – Dataset containing at least `'ISF_mask'`, `'GL_mask'`, `'IF_mask'`

Returns
whole_ds – whole_ds extended with the variables `'dGL'`, `'dIF'` and `'dGL_dIF'`

Return type
xr.Dataset

multimelt.create_isf_mask_functions.**create_isf_masks**(*file_map*, *file_msk*, *file_conc*, *xx*, *yy*, *latlonboundary_file*, *outputpath*, *chunked*, *dx*, *dy*, *FRIS_one=True*, *mouginot_basins=False*, *variable_geometry=False*, *ground_point='yes'*, *write_ismask='yes'*, *write_groundmask='yes'*, *dist=150*, *add_fac=100*, *connectivity=4*, *threshold=4*)

Identify the location of ice shelves, Antarctic mainland, grounding lines, ice fronts and pinning points.

This function creates masks identifying the location of ice shelves, Antarctic mainland, grounding lines, ice fronts and pinning points

Parameters

- **file_map** (*xr.Dataset*) – Dataset containing information about the grid
- **file_msk** (*xr.DataArray*) – Mask separating ocean (0), ice shelves (between 0 and 2, excluding 0 and 2), grounded ice (2)
- **file_conc** (*xr.DataArray*) – Ice shelf concentration for each point (between 0 and 1)
- **xx** (*xr.DataArray*) – x-coordinates for the domain.
- **yy** (*xr.DataArray*) – y-coordinates for the domain.
- **latlonboundary_file** (*str*) – Path to the csv-file containing the info. This function is tailored to the format of `lonlat_masks.txt`. It takes the path to a netcdf-file if mouginot_basins is True.
- **outputpath** (*str*) – Path where the intermediate masks should be written to.
- **chunked** (*int or False*) – Size of chunks for dask when opening a netcdf into a dataset, if no need to chunk: False.
- **dx** (*float*) – Grid size in x direction, step from left to right (can be positive or negative depending on the initial coordinate).
- **dy** (*float*) – Grid size in x direction, step from left to right (can be positive or negative depending on the initial coordinate).
- **FRIS_one** (*boolean*) – True if Filchner-Ronne should be treated as one ice shelf, False if Filchner and Ronne should be separated.
- **mouginot_basins** (*Boolean*) – If True, arr_def_ismask
- **ground_point** (*str*) – `yes` or `no`. If `yes`, the grounding line is defined on the ground points at the border to the ice shelf. If `no`, the grounding line is defined on the ice shelf points at the border to the ground.
- **write_ismask** (*str*) – `yes` or `no`. If `yes`, compute the mask of the different ice shelves. If `no`, read in the already existing file `outputpath + 'preliminary_mask_file.nc'`.
- **write_groundmask** (*str*) – `yes` or `no`. If `yes`, compute the mask of mainland Antarctica. If `no`, read in the already existing file `outputpath + 'mask_ground.nc'`.
- **dist** (*int*) – Defines the size of the starting square for the ground mask - should be small if the resolution is coarse and high if the resolution is fine. Default is currently 150 but you can play around. A good indicator to see if it is too high is if you see the small upper tail of the Ross ice shelf or if it is masked as ground.
- **add_fac** (*int*) – Defines additional iterations for the propagation for the ground mask. Was introduced to get to the end of the Antarctic Peninsula, sometimes it would not get there otherwise. Current default is 100 but you are welcome to play around with it.
- **connectivity** (*int*) – 4 or 8 for 2D, defines what is considered a "connected" point when looking at ice-shelves
- **threshold** (*int*) – Size of lonely pixel areas to remove for ice-shelf mask.

Returns

- **outfile** (*xr.Dataset*) – Dataset containing all the produced masks: ISF_mask, GL_mask, IF_mask, PP_mask, ground_mask
- **new_mask** (*xr.DataArray*) – Array showing the coverage of each ice shelf with the respective ID, open ocean is 1, land is 0.
- **mask_gline** (*xr.DataArray*) – Array showing the grounding line with the ID of the corresponding ice shelf.
- **mask_front** (*xr.DataArray*) – Array showing the ice front with the ID of the corresponding ice shelf.

multimelt.create_isf_mask_functions.**create_mask_and_metadata_isf**(*file_map*, *file_bed*, *file_msk*, *file_draft*, *file_conc*, *chunked*, *latlonboundary_file*, *outputpath*, *file_metadata*, *file_metadata_GL_flux*, *ground_point*, *FRIS_one=True*, *mouginot_basins=False*, *variable_geometry=False*, *write_ismask='yes'*, *write_groundmask='yes'*, *write_outfile='yes'*, *dist=150*, *add_fac=100*, *connectivity=4*, *threshold=4*, *write_metadata='yes'*)

Create mask and metadata file for all ice shelves.

This function creates a dataset containing masks and metadata for the ice shelves in Antarctica. The input must be on a stereographic grid centered around Antarctica.

Parameters

- **file_map** (*xr.Dataset*) – Dataset containing information about the grid
- **file_bed** (*xr.DataArray*) – Array containing the bedrock topography at least at each ocean/ice shelf point. Bedrock depth should be negative when below sea level!
- **file_msk** (*xr.DataArray*) – Mask separating ocean (0), ice shelves (3), grounded ice (2) (also can contain ice free land (1), and lake Vostok (4)).
- **file_draft** (*xr.DataArray*) – Array containing the ice draft depth at at least at each ocean/ice shelf. Ice draft depth should be negative when below sea level!
- **file_conc** (*float between 0 and 1*) – Concentration of ice shelf in each grid cell
- **chunked** (*int or False*) – Size of chunks for dask when opening a netcdf into a dataset, if no need to chunk: False.
- **latlonboundary_file** (*str*) – Path to the csv-file containing the info. This function is tailored to the format of `lonlat_masks.txt`.
- **outputpath** (*str*) – Path where the intermediate files should be written to.
- **file_metadata** (*str*) – Path to `iceshelves_metadata_Nico.txt`
- **file_metadata_GL_flux** (*str*) – Path to `GL_flux_rignot13.csv`
- **ground_point** (*str*) – `yes` or `no`. If `yes`, the grounding line is defined on the ground points at the border to the ice shelf. If `no`, the grounding line is defined on the ice shelf points at the border to the ground.

- **FRIS_one** (*boolean*) – True if Filchner-Ronne should be treated as one ice shelf, False if Filchner and Ronne should be separated.
- **write_ismask** (*str*) – yes or no. If yes, compute the mask of the different ice shelves. If no, read in the already existing file outputpath + 'preliminary_mask_file.nc'.
- **write_groundmask** (*str*) – yes or no. If yes, compute the mask of mainland Antarctica. If no, read in the already existing file outputpath + 'mask_ground.nc'.
- **write_outfile** (*str*) – yes or no. If yes, go through the mask file. If no, read in the already existing file outputpath + 'outfile.nc'.
- **dist** (*int*) – Defines the size of the starting square for the ground mask - should be small if the resolution is coarse and high if the resolution is fine. Default is currently 150 but you can play around. A good indicator to see if it is too high is if you see the small upper tail of the Ross ice shelf or if it is masked as ground.
- **add_fac** (*int*) – Defines additional iterations for the propagation for the ground mask. Was introduced to get to the end of the Antarctic Peninsula, sometimes it would not get there otherwise. Current default is 100 but you are welcome to play around with it.
- **write_metadata** (*str*) – yes or no. If yes, prepare the metadata csv-file. If no, read in the already existing file outputpath + 'ice_shelf_metadata_complete.csv'.

Returns
whole_ds – Dataset summarizing all masks and info needed to prepare and use the parametrizations.

Return type
xr.Dataset

multimelt.create_isf_mask_functions.**def_ground_mask**(*file_msk*, *dist*, *add_fac*)

Define a mask for the Antarctic continent as such (not the islands).

This function defines the points that are part of the Antarctic continent as such (not the islands).

Parameters

- **file_msk** (*xr.DataArray*) – Mask separating ocean (0), ice shelves (between 0 and 2, excluding 0 and 2), grounded ice (2)
- **dist** (*int*) – Defines the size of the starting square - should be small if the resolution is coarse and high if the resolution is fine. Default is currently 150 but you can play around. A good indicator to see if it is too high is if you see the small upper tail of the Ross ice shelf or if it is masked as ground.
- **add_fac** (*int*) – Defines additional iterations. Was introduced to get to the end of the Antarctic Peninsula, sometimes it would not get there otherwise. Current default is 100 but you are welcome to play around with it.

Returns
mask_ground – Array showing the coverage of the Antarctic continent (0 for islands, 1 for ocean and ice shelves, 2 for mainland)

Return type
xr.DataArray

multimelt.create_isf_mask_functions.**def_grounding_line**(*new_mask*, *mask_ground*, *ground_point*, *add_fac*, *dx*, *dy*)

Identify grounding line points and assign ice shelf ID to these points.

This function draws the grounding line of the different ice shelves. You can decide if they are the points on the ground or on the ice shelf side. CAREFUL: I would recommend you choose setting ground_point = 'no'. In that case Alexander Island is taken into account. I have not been able to arrange that for ground_point = 'yes'.

Parameters

- **new_mask** (*xr.DataArray*) – Array showing the coverage of each ice shelf with the respective ID, open ocean is 1, land is 0.
- **mask_ground** (*xr.DataArray*) – Array showing the coverage "mainland" Antarctica. Mainland is 2, islands are 1, all else is 0.
- **ground_point** (*str*) – `yes` or `no`. If `yes`, the grounding line is defined on the ground points at the border to the ice shelf. If `no`, the grounding line is defined on the ice shelf points at the border to the ground.
- **add_fac** (*int*) – Defines additional iterations for the propagation for the ground mask. Was introduced to get to the end of the Antarctic Peninsula, sometimes it would not get there otherwise. Now useful to also propagate ground in Alexander Island.
- **dx** (*float*) – Grid size in x direction, step from left to right (can be positive or negative depending on the initial coordinate).
- **dy** (*float*) – Grid size in x direction, step from left to right (can be positive or negative depending on the initial coordinate).

Returns
mask_gline_final – Array showing the grounding line with the ID of the corresponding ice shelf.

Return type
xr.DataArray

multimelt.create_isf_mask_functions.**def_ice_front**(*new_mask*, *file_msk*)

Identify ice front points and assign ice shelf ID to these points.

This function draws the ice front of the different ice shelves. They are the points on the ice shelf side.

Parameters

- **new_mask** (*xr.DataArray*) – Array showing the coverage of each ice shelf with the respective ID, open ocean is 1, land is 0.
- **file_msk** (*xr.DataArray*) – Mask separating ocean (0), ice shelves (between 0 and 2, excluding 0 and 2), grounded ice (2)

Returns
mask_front – Array showing the ice front with the ID of the corresponding ice shelf.

Return type
xr.DataArray

multimelt.create_isf_mask_functions.**def_isf_mask**(*arr_def_ismask*, *file_msk*, *file_conc*, *lon*, *lat*, *FRIS_one=True*, *mouginot_basins=False*, *variable_geometry=False*, *connectivity=4*, *threshold=4*)

Define a mask for the individual ice shelves.

This function defines a mask for the individual ice shelves. I think it works for both stereographic and latlon grids but I have not tried the latter.

Parameters

- **arr_def_ismask** (*np.array*) – Array containing minlon,maxlon,minlat,maxlat,is_nb or xr.Dataset with drainage basins
- **file_msk** (*xr.DataArray*) – Mask separating ocean (0), ice shelves (between 0 and 2, excluding 0 and 2), grounded ice (2)
- **file_conc** (*xr.DataArray*) – Ice shelf concentration for each point (between 0 and 1)
- **lon** (*xr.DataArray*) – Longitude (depends on x,y for stereographic)
- **lat** (*xr.DataArray*) – Latitude (depends on x,y for stereographic)
- **FRIS_one** (*Boolean*) – If True, Filchner-Ronne are considered as one ice-shelf
- **mouginot_basins** (*Boolean*) – If True, arr_def_ismask is an xr.DataArray with drainage basins
- **variable_geometry** (*Boolean*) – If True, arr_def_ismask
- **connectivity** (*int*) – 4 or 8 for 2D, defines what is considered a "connected" point
- **threshold** (*int*) – Size of lonely pixel areas to remove

Returns
new_mask – Array showing the coverage of each ice shelf with the respective ID, open ocean is 1, land is 0

Return type
xr.DataArray

multimelt.create_isf_mask_functions.**def_pinning_point_boundaries**(*mask_pin*, *new_mask*)

Identify the boundaries of pinning points and assign the negative value of the ice shelf ID to these points.

This function draws the boundaries of the pinning points of the different ice shelves.

Parameters

- **mask_pin** (*xr.DataArray*) – Array showing the pinning points with the negative value of the ID of the corresponding ice shelf.
- **new_mask** (*xr.DataArray*) – Array showing the coverage of each ice shelf with the respective ID, open ocean is 1, land is 0.

Returns
mask_pin2 – Array showing the boundaries of the pinning points with the negative value of the ID of the corresponding ice shelf.

Return type
xr.DataArray

multimelt.create_isf_mask_functions.**def_pinning_points**(*new_mask*, *lon*, *lat*, *mask_ground*)

Identify pinning points and assign the negative value of the ice shelf ID to these points.

This function identifies islands within or at the limit of the different ice shelves.

Parameters

- **new_mask** (*xr.DataArray*) – Array showing the coverage of each ice shelf with the respective ID, open ocean is 1, land is 0.
- **lon** (*xr.DataArray*) – Longitude (depends on x,y for stereographic)
- **lat** (*xr.DataArray*) – Latitude (depends on x,y for stereographic)
- **mask_ground** (*xr.DataArray*) – Array showing the coverage of the Antarctic continent (0 for islands, 1 for ocean and ice shelves, 2 for mainland)

Returns
mask_pin – Array showing the pinning points with the negative value of the ID of the corresponding ice shelf.

Return type
xr.DataArray

multimelt.create_isf_mask_functions.**prepare_csv_metadata**(*file_metadata*, *file_metadata_GL_flux*, *file_conc*, *dx*, *dy*, *new_mask*, *FRIS_one*, *mouginot_basins*)

Prepare the metadata info (ice shelf names and observed melt rates).

This function creates a dataframe with the metadata for each ice shelf (name, area, observed melt rates from Rignot)

Parameters

- **file_metadata** (*str*) – Path to `iceshelves_metadata_Nico.txt`
- **file_metadata_GL_flux** (*str*) – Path to `GL_flux_rignot13.csv`
- **file_conc** (*float between 0 and 1*) – Concentration of ice shelf in each grid cell
- **dx** (*float*) – Grid spacing in the x-direction
- **dy** (*float*) – Grid spacing in the y-direction
- **new_mask** (*xr.DataArray*) – Array showing the coverage of each ice shelf with the respective ID, open ocean is 1, land is 0.
- **FRIS_one** (*boolean*) – True if Filchner-Ronne should be treated as one ice shelf, False if Filchner and Ronne should be separated.

Returns
df1 – DataFrame containing the following columns for each ice shelf: columns=['isf_name', 'region', 'isf_melt', 'melt_uncertainty','isf_area_rignot']

Return type
pandas.DataFrame

multimelt.create_isf_mask_functions.**prepare_metadata**(*file_metadata*, *file_metadata_GL_flux*, *dx*, *dy*, *new_mask*, *mask_gline*, *mask_front*, *file_draft*, *file_bed*, *file_conc*, *lon*, *lat*, *outputpath*, *write_metadata='yes'*, *FRIS_one=True*, *mouginot_basins=True*)

Prepare the metadata info into a csv.

This function creates a panda DataFrame with all metadata (1D) info about the individual ice shelves.

Parameters

- **file_metadata** (*str*) – Path to `iceshelves_metadata_Nico.txt`
- **file_metadata_GL_flux** (*str*) – Path to `GL_flux_rignot13.csv`
- **dx** (*float*) – Grid spacing in the x-direction
- **dy** (*float*) – Grid spacing in the y-direction
- **new_mask** (*xr.DataArray*) – Array showing the coverage of each ice shelf with the respective ID, open ocean is 1, land is 0.
- **mask_gline** (*xr.DataArray*) – Array showing the grounding line with the ID of the corresponding ice shelf.

- **mask_front** (*xr.DataArray*) – Array showing the ice front with the ID of the corresponding ice shelf.
- **file_draft** (*xr.DataArray*) – Array containing the ice draft depth at at least at each ocean/ice shelf. Ice draft depth should be negative when below sea level!
- **file_bed** (*xr.DataArray*) – Array containing the bedrock topography at least at each ocean/ice shelf point. Bedrock depth should be negative when below sea level!
- **file_conc** (*xr.DataArray, float between 0 and 1*) – Concentration of ice shelf in each grid cell
- **lon** (*xr.DataArray*) – Longitude (depends on x,y for stereographic)
- **lat** (*xr.DataArray*) – Latitude (depends on x,y for stereographic)
- **outputpath** (*str*) – Path where the metadata csv-file should be written to.
- **write_metadata** (*str*) – `yes` or `no`. If `yes`, prepare the metadata csv-file. If `no`, read in the already existing file `outputpath + 'ice_shelf_metadata_complete.csv'`.
- **FRIS_one** (*boolean*) – True if Filchner-Ronne should be treated as one ice shelf, False if Filchner and Ronne should be separated.

Returns

- **df1** (*pandas.DataFrame*) – DataFrame containing the following columns for each ice shelf: columns=['isf_name', 'region', 'isf_melt', 'melt_uncertainty','isf_area_rignot','front_bot_depth_max','front_bot_depth_avg','front_ice_depth_min','front_ice
- **ds1** (*xr.Dataset*) – xarray Dataset containing ['front_bot_depth_max','front_bot_depth_avg','front_ice_depth_min',' if time dependent geometry.

multimelt.create_isf_mask_functions.**read_isfmask_info**(*infile*)

Read the ice shelf limits and info from infile and convert it to an array.

This function reads the ice shelf limits (min/max lon, min/max lat, ice shelf ID) from infile and coverts it to a np.array. This function is tailored to the format of `lonlat_masks.txt`.

Parameters

infile (*str*) – path to the csv-file containing the info. This function is tailored to the format of `lonlat_masks.txt`.

Returns

res – array containing minlon,maxlon,minlat,maxlat,is_nb

Return type

np.array

1.6.6 melt_functions

multimelt.melt_functions.**PICOP_param**(*T_in*, *S_in*, *box_location*, *box_depth_below_surface*, *box_area_whole*, *nD*, *spatial_coord*, *isf_cell_area*, *gamma_pico*, *gamma_plume*, *E0*, *C*, *zGL*, *alpha*, *ice_draft_neg*, *pism_version='no'*, *picop_opt='2019'*)

Compute melt rate using PICOP.

This function computes the basal melt based on PICOP (see Pelle et al., 2019). Using the empirical equations from Lazeroms et al. 2018.

Parameters

- **T_in** (*xr.DataArray*) – Temperature entering the cavity in degrees C.
- **S_in** (*xr.DataArray*) – Salinity entering the cavity in psu.
- **box_location** (*xr.DataArray*) – Spatial location of the boxes (dims=['box_nb_tot']),
- **box_depth_below_surface** (*xr.DataArray*) – Mean ice draft depth of the box in m (dims=['box_nb_tot','box_nb']). Negative downwards!
- **box_area_whole** (*xr.DataArray*) – Area of the different boxes (dims=['box_nb_tot','box_nb']).
- **nD** (*scalar*) – Number of boxes to use (i.e. box_nb_tot).
- **spatial_coord** (*list of str*) – Coordinate(s) to use for spatial means.
- **isf_cell_area** (*float*) – Area covered by ice shelf in each cell
- **gamma_pico** (*float*) – Gamma to be used in PICO part.
- **gamma_plume** (*float*) – Gamma to be used in plume part.
- **E0** (*float*) – Entrainment coefficient.
- **C** (*float*) – Circulation parameter C (Sv m3 kg-1 = m6 kg-1 s-1) in [0.1;9]*1.e6.
- **pism_version** (*str*) – Can be `yes` or `no`, depending on if you want to use the PICO version as implemented in PISM or the original PICO (uniform box melt). See Sec. 2.4 by Reese et al. 2018 for more info.
- **zGL** (*scalar or array*) – Depth of the grounding line where the source of the plume is in m (depth is negative!).
- **alpha** (*scalar or array*) – Slope angle in rad (must be positive).
- **ice_draft_neg** (*scalar or array*) – Depth of the ice draft in m (depth is negative!).
- **pism_version** – Can be `yes` or `no`, depending on if you want to use the PICO version as implemented in PISM or the original PICO (uniform box melt). See Sec. 2.4 by Reese et al. 2018 for more info.
- **picop_opt** (*str*) – Can be `2019`, `2018` , depending on if you want to use PICOP with analytical plume param or with empirical plume param.

Returns

- **T_all_boxes** (*xr.DataArray*) – Temperature field in degrees C.
- **S_all_boxes** (*xr.DataArray*) – Salinity field in psu.
- **m_all_boxes** (*xr.DataArray*) – Melt rate field in m ice per second.

multimelt.melt_functions.**PICO_and_PICOP_param**(*T_in*, *S_in*, *box_location*, *box_depth_below_surface*, *box_area_whole*, *nD*, *spatial_coord*, *isf_cell_area*, *gamma*, *E0*, *C*, *picop='no'*, *pism_version='no'*, *zGL=None*, *alpha=None*, *ice_draft_neg=None*)

Compute melt rate using PICO or PICOP. THIS FUNCTION IS NOT USED ANYMORE => NOW TWO SEPARATE FOR PICO AND PICOP

This function computes the basal melt based on PICO or PICOP (see Reese et al., 2018 and Pelle et al., 2019).

Parameters

- **T_in** (*xr.DataArray*) – Temperature entering the cavity in degrees C.
- **S_in** (*xr.DataArray*) – Salinity entering the cavity in psu.

- **box_location** (*xr.DataArray*) – Spatial location of the boxes (dims=['box_nb_tot']),
- **box_depth_below_surface** (*xr.DataArray*) – Mean ice draft depth of the box in m (dims=['box_nb_tot','box_nb']). Negative downwards!
- **box_area_whole** (*xr.DataArray*) – Area of the different boxes (dims=['box_nb_tot','box_nb']).
- **nD** (*scalar*) – Number of boxes to use (i.e. box_nb_tot).
- **spatial_coord** (*list of str*) – Coordinate(s) to use for spatial means.
- **isf_cell_area** (*float*) – Area covered by ice shelf in each cell
- **gamma** (*float*) – Gamma to be tuned in m/s.
- **E0** (*float*) – Entrainment coefficient.
- **C** (*float*) – Circulation parameter C (Sv m3 kg-1 = m6 kg-1 s-1) in [0.1;9]*1.e6.
- **picop** (*str*) – Can be `yes` or `no`, depending on if you want to use PICOP or the original PICO.
- **pism_version** (*str*) – Can be `yes` or `no`, depending on if you want to use the PICO version as implemented in PISM or the original PICO (uniform box melt). See Sec. 2.4 by Reese et al. 2018 for more info.
- **zGL** (*scalar or array*) – Depth of the grounding line where the source of the plume is in m (depth is negative!).
- **alpha** (*scalar or array*) – Slope angle in rad (must be positive).
- **ice_draft_neg** (*scalar or array*) – Depth of the ice draft in m (depth is negative!).#

Returns

- **T_all_boxes** (*xr.DataArray*) – Temperature field in degrees C.
- **S_all_boxes** (*xr.DataArray*) – Salinity field in psu.
- **m_all_boxes** (*xr.DataArray*) – Melt rate field in m ice per second.

multimelt.melt_functions.**PICO_param**(*T_in*, *S_in*, *box_location*, *box_depth_below_surface*, *box_area_whole*, *nD*, *spatial_coord*, *isf_cell_area*, *gamma*, *C*, *ice_draft_neg*, *pism_version='no'*)

Compute melt rate using PICO.

This function computes the basal melt based on PICO (see Reese et al., 2018).

Parameters

- **T_in** (*xr.DataArray*) – Temperature entering the cavity in degrees C.
- **S_in** (*xr.DataArray*) – Salinity entering the cavity in psu.
- **box_location** (*xr.DataArray*) – Spatial location of the boxes (dims=['box_nb_tot']),
- **box_depth_below_surface** (*xr.DataArray*) – Mean ice draft depth of the box in m (dims=['box_nb_tot','box_nb']). Negative downwards!
- **box_area_whole** (*xr.DataArray*) – Area of the different boxes (dims=['box_nb_tot','box_nb']).
- **nD** (*scalar*) – Number of boxes to use (i.e. box_nb_tot).
- **spatial_coord** (*list of str*) – Coordinate(s) to use for spatial means.
- **isf_cell_area** (*float*) – Area covered by ice shelf in each cell

- **gamma** (*float*) – Gamma to be tuned in m/s.
- **C** (*float*) – Circulation parameter C (Sv m3 kg-1 = m6 kg-1 s-1) in [0.1;9]*1.e6.
- **ice_draft_neg** (*scalar or array*) – Depth of the ice draft in m (depth is negative!).
- **pism_version** (*str*) – Can be `yes` or `no`, depending on if you want to use the PICO version as implemented in PISM or the original PICO (uniform box melt). See Sec. 2.4 by Reese et al. 2018 for more info.

Returns

- **T_all_boxes** (*xr.DataArray*) – Temperature field in degrees C.
- **S_all_boxes** (*xr.DataArray*) – Salinity field in psu.
- **m_all_boxes** (*xr.DataArray*) – Melt rate field in m ice per second.

multimelt.melt_functions.**T_correction_PICO**(*isf_name*, *T_in*)

Apply regional temperature corrections to input temperatures for box model.

This functions produces locally "corrected" temperature profiles for input to box model, fitting the "new" Reese parameters.

Parameters

- **isf_name** (*str*) – Name of the ice shelf of interest.
- **T_in** (*array*) – Temperature profile in degrees C.

Returns
T_corrected – Corrected temperature profile in degrees C.

Return type
array

multimelt.melt_functions.**calculate_melt_rate_1D_all_isf**(*nisf_list*, *ds_melt_rate_2D_all*, *geometry_info_2D*, *isf_stack_mask*, *options_1D=['melt_m_ice_per_y_avg', 'melt_m_ice_per_y_min', 'melt_m_ice_per_y_max', 'melt_we_per_y_tot', 'melt_we_per_y_avg', 'melt_Gt_per_y_tot']*, *verbose=True*)

Function to transform 2D melt information into 1D values.

Parameters

- **nisf_list** (*array of int*) – List containing the ice shelf IDs for all ice shelves of interest.
- **ds_melt_rate_2D_all** (*xarray.Dataset*) – Dataset containing 2D melt information.
- **geometry_info_2D** (*xarray.Dataset*) – Dataset containing relevant 2D geometrical information.
- **isf_stack_mask** (*xarray.DataArray*) – DataArray containing the stacked coordinates of the ice shelves (to make computing faster).
- **options_1D** (*list of str*) – 1D variables to be written out. Possible options: 'melt_m_ice_per_y_avg', 'melt_m_ice_per_y_min', 'melt_m_ice_per_y_max', 'melt_we_per_y_tot'. 'melt_m_ice_per_y_tot' is always written out!
- **verbose** (*Boolean*) – `True` if you want the program to keep you posted on where it is in the calculation.

Returns
ds_melt_rates_1D_tot – 1D metrics about the melt rate for all ice shelves.

Return type
xarray.Dataset

multimelt.melt_functions.**calculate_melt_rate_1D_and_2D_all_isf**(*nisf_list*, *T_S_profile*, *geometry_info_2D*, *geometry_info_1D*, *isf_stack_mask*, *mparam*, *gamma*, *U_param=True*, *C=None*, *E0=None*, *angle_option='lazero'*, *box_charac_2D=None*, *box_charac_1D=None*, *box_tot=None*, *box_tot_option='box_nb_tot'*, *pism_version='no'*, *picop_opt='no'*, *gamma_plume=None*, *T_corrections=False*, *options_2D=['melt_m_ice_per_y', 'melt_m_we_per_y']*, *options_1D=['melt_m_ice_per_y_avg', 'melt_m_ice_per_y_min', 'melt_m_ice_per_y_max', 'melt_we_per_y_tot', 'melt_we_per_y_avg', 'melt_Gt_per_y_tot']*, *HUB=False*, *verbose=True*)

Function to process input information and call the 2D and 1D functions to compute melt rate variables.

Parameters

- **nisf_list** (*array of int*) – List containing the ice shelf IDs for all ice shelves of interest.
- **T_S_profile** (*xarray.Dataset*) – Dataset containing temperature (in degrees C) and salinity (in psu) input profiles.
- **geometry_info_2D** (*xarray.Dataset*) – Dataset containing relevant 2D geometrical information.
- **geometry_info_1D** (*xarray.Dataset*) – Dataset containing relevant 1D geometrical information.
- **isf_stack_mask** (*xarray.DataArray*) – DataArray containing the stacked coordinates of the ice shelves (to make computing faster).
- **mparam** (*str*) – Parameterisation to be applied.
- **gamma** (*float*) – Gamma to be tuned.
- **U_param** (*Boolean*) – If `True` we use the complex parameterisation of U, if `False`, this is "only" equal to (rho_sw * c_po) / (rho_i * L_i). Relevant for simple parameterisations only.
- **C** (*float*) – Circulation parameter C (Sv m3 kg-1 = m6 kg-1 s-1) in [0.1;9]*1.e6.
- **E0** (*float*) – Entrainment coefficient.
- **angle_option** (*str*) – Slope to be used, choice between "cavity" (cavity), "lazero" (lazeroms18), "local" (local)

- **box_charac_2D** (*xarray.Dataset*) – Dataset containing relevant 2D box characteristics for all ice shelves.
- **box_charac_1D** (*xarray.Dataset*) – Dataset containing relevant 1D box characteristics for all ice shelves.
- **box_tot** (*int*) – Either the total number of boxes being used if box_tot_option='box_nb_tot' or the configuration to use if box_tot_option='nD_config'.
- **box_tot_option** (*str*) – Defines how box_tot``should be interpreted. Can be either 'box_nb_tot', then ``box_tot is the total number of boxes or 'nD_config', then box_tot is the configuration to use.
- **pism_version** (*str*) – Can be yes or no, depending on if you want to use the PICO version as implemented in PISM or the original PICO (uniform box melt). See Sec. 2.4 by Reese et al. 2018 for more info.
- **picop_opt** (*str*) – Can be yes or no, depending on if you want to use PICOP or the original PICO.
- **gamma_plume** (*float*) – Gamma to be used in the plume part of PICOP.
- **T_corrections** (*Boolean*) – If True, use regional corrections (preferentially only when using "new" Reese parameters).
- **options_2D** (*list of str*) – 2D variables to be written out. Possible options: 'melt_m_ice_per_y','melt_m_we_per_y'. 'melt_m_ice_per_s' is always written out!
- **options_1D** (*list of str*) – 1D variables to be written out. Possible options: 'melt_m_ice_per_y_avg', 'melt_m_ice_per_y_min', 'melt_m_ice_per_y_max', 'melt_we_per_y_tot'. 'melt_m_ice_per_y_tot' is always written out!
- **verbose** (*Boolean*) – True if you want the program to keep you posted on where it is in the calculation.

Returns

- **ds_2D** (*xarray.Dataset*) – Horizontal distribution of the melt rate for all ice shelves in one map.
- **ds_1D** (*xarray.Dataset*) – 1D metrics about the melt rate for all ice shelves.

multimelt.melt_functions.**calculate_melt_rate_2D_1isf**(*kisf*, *T_S_profile*, *geometry_info_2D*, *geometry_info_1D*, *isf_stack_mask*, *mparam*, *gamma*, *U_param=True*, *C=None*, *E0=None*, *angle_option='lazero'*, *box_charac_2D=None*, *box_charac_1D=None*, *box_tot=None*, *box_tot_option='box_nb_tot'*, *pism_version='no'*, *picop_opt='no'*, *gamma_plume=None*, *T_corrections=False*, *HUB=False*)

Wrap function to point to the right melt parameterisation for one ice shelf.

Parameters

- **kisf** (*int*) – Ice shelf ID for the ice shelf of interest.
- **T_S_profile** (*xarray.Dataset*) – Dataset containing temperature (in degrees C) and salinity (in psu) input profiles.
- **geometry_info_2D** (*xarray.Dataset*) – Dataset containing relevant 2D geometrical information.

- **geometry_info_1D** (*xarray.Dataset*) – Dataset containing relevant 1D geometrical information.
- **isf_stack_mask** (*xarray.DataArray*) – DataArray containing the stacked coordinates of the ice shelves (to make computing faster).
- **mparam** (*str*) – Parameterisation to be applied.
- **gamma** (*float*) – Gamma to be tuned. Will be used as gamma_pico in PICOP.
- **U_param** (*Boolean*) – If `True` we use the complex parameterisation of U, if `False`, this is "only" equal to (rho_sw * c_po) / (rho_i * L_i). Relevant for simple parameterisations only.
- **C** (*float*) – Circulation parameter C (Sv m3 kg-1 = m6 kg-1 s-1) in [0.1;9]*1.e6.
- **E0** (*float*) – Entrainment coefficient.
- **angle_option** (*str*) – Slope to be used, choice between "cavity" (cavity slope), "lazero" (lazeroms18), "local" (local slope)
- **box_charac_2D** (*xarray.Dataset*) – Dataset containing relevant 2D box characteristics for all ice shelves.
- **box_charac_1D** (*xarray.Dataset*) – Dataset containing relevant 1D box characteristics for all ice shelves.
- **box_tot** (*int*) – Either the total number of boxes being used if box_tot_option='box_nb_tot' or the configuration to use if box_tot_option='nD_config'.
- **box_tot_option** (*str*) – Defines how `box_tot``should be interpreted. Can be either 'box_nb_tot', then ``box_tot` is the total number of boxes or 'nD_config', then `box_tot` is the configuration to use.
- **pism_version** (*str*) – Can be `yes` or `no`, depending on if you want to use the PICO version as implemented in PISM or the original PICO (uniform box melt). See Sec. 2.4 by Reese et al. 2018 for more info.
- **picop_opt** (*str*) – Can be `2019`, `2018` or `no`, depending on if you want to use PICOP with analytical plume param, with empirical plume param or the original PICO without plume.
- **gamma_plume** (*float*) – Gamma to be used in the plume part of PICOP.
- **T_corrections** (*Boolean*) – If `True`, use regional corrections (preferentially only when using "new" Reese parameters).

Returns

melt_rate_2D_isf – Horizontal distribution of the melt rate in m ice/s for the ice shelf of interest.

Return type

xr.DataArray

multimelt.melt_functions.**calculate_melt_rate_2D_all_isf**(*nisf_list*, *T_S_profile*, *geometry_info_2D*, *geometry_info_1D*, *isf_stack_mask*, *mparam*, *gamma*, *U_param=True*, *C=None*, *E0=None*, *angle_option='lazero'*, *box_charac_2D=None*, *box_charac_1D=None*, *box_tot=None*, *box_tot_option='box_nb_tot'*, *pism_version='no'*, *picop_opt='no'*, *gamma_plume=None*, *T_corrections=False*, *options_2D=['melt_m_ice_per_y', 'melt_m_we_per_y']*, *HUB=False*, *verbose=True*)

Wrap function to loop over all ice shelves and combine result to a map.

Parameters

- **nisf_list** (*array of int*) – List containing the ice shelf IDs for all ice shelves of interest.
- **T_S_profile** (*xarray.Dataset*) – Dataset containing temperature (in degrees C) and salinity (in psu) input profiles.
- **geometry_info_2D** (*xarray.Dataset*) – Dataset containing relevant 2D geometrical information.
- **geometry_info_1D** (*xarray.Dataset*) – Dataset containing relevant 1D geometrical information.
- **isf_stack_mask** (*xarray.DataArray*) – DataArray containing the stacked coordinates of the ice shelves (to make computing faster).
- **mparam** (*str*) – Parameterisation to be applied.
- **gamma** (*float*) – Gamma to be tuned.
- **U_param** (*Boolean*) – If `True` we use the complex parameterisation of U, if `False`, this is "only" equal to (rho_sw * c_po) / (rho_i * L_i). Relevant for simple parameterisations only.
- **C** (*float*) – Circulation parameter C (Sv m3 kg-1 = m6 kg-1 s-1) in [0.1;9]*1.e6.
- **E0** (*float*) – Entrainment coefficient.
- **angle_option** (*str*) – Slope to be used, choice between "cavity" (cavity), "lazero" (lazeroms18), "local" (local)
- **box_charac_2D** (*xarray.Dataset*) – Dataset containing relevant 2D box characteristics for all ice shelves.
- **box_charac_1D** (*xarray.Dataset*) – Dataset containing relevant 1D box characteristics for all ice shelves.
- **box_tot** (*int*) – Either the total number of boxes being used if box_tot_option='box_nb_tot' or the configuration to use if box_tot_option='nD_config'.
- **box_tot_option** (*str*) – Defines how `box_tot``should be interpreted. Can be either 'box_nb_tot', then ``box_tot` is the total number of boxes or 'nD_config', then `box_tot` is the configuration to use.
- **pism_version** (*str*) – Can be `yes` or `no`, depending on if you want to use the PICO version as implemented in PISM or the original PICO (uniform box melt). See Sec. 2.4 by Reese et al. 2018 for more info.
- **picop_opt** (*str*) – Can be `yes` or `no`, depending on if you want to use PICOP or the original PICO.
- **gamma_plume** (*float*) – Gamma to be used in the plume part of PICOP.
- **T_corrections** (*Boolean*) – If `True`, use regional corrections (preferentially only when using "new" Reese parameters).
- **options_2D** (*list of str*) – 2D variables to be written out. Possible options: 'melt_m_ice_per_y','melt_m_we_per_y'. 'melt_m_ice_per_s' is always written out!
- **verbose** (*Boolean*) – `True` if you want the program to keep you posted on where it is in the calculation.

Returns
ds_melt_rate_2D_unstacked – Horizontal distribution of the melt rate for all ice shelves in one map.

Return type
xarray.Dataset

multimelt.melt_functions.**calculate_melt_rate_2D_boxes_1isf**(*kisf*, *T_S_profile*, *geometry_info_2D*, *geometry_info_1D*, *box_charac_all_2D*, *box_charac_all_1D*, *isf_stack_mask*, *mparam*, *box_tot_nb*, *gamma*, *C*, *angle_option*, *pism_version*, *T_corrections*)

Function to compute melt from box parameterisations for one ice shelf.

Parameters

- **kisf** (*int*) – Ice shelf ID for the ice shelf of interest.
- **T_S_profile** (*xarray.Dataset*) – Dataset containing temperature (in degrees C) and salinity (in psu) input profiles.
- **geometry_info_2D** (*xarray.Dataset*) – Dataset containing relevant 2D geometrical information.
- **geometry_info_1D** (*xarray.Dataset*) – Dataset containing relevant 1D geometrical information.
- **box_charac_all_2D** (*xarray.Dataset*) – Dataset containing relevant 2D box characteristics for all ice shelves.
- **box_charac_all_1D** (*xarray.Dataset*) – Dataset containing relevant 1D box characteristics for all ice shelves.
- **isf_stack_mask** (*xarray.DataArray*) – DataArray containing the stacked coordinates of the ice shelves (to make computing faster).
- **mparam** (*str*) – Parameterisation to be applied.
- **box_tot_nb** (*int*) – Total number of boxes
- **gamma** (*float*) – Gamma to be tuned.
- **C** (*float*) – Circulation parameter C (Sv m3 kg-1 = m6 kg-1 s-1) in [0.1;9]*1.e6.
- **pism_version** (*str*) – Can be `yes` or `no`, depending on if you want to use the PICO version as implemented in PISM or the original PICO (uniform box melt). See Sec. 2.4 by Reese et al. 2018 for more info.
- **T_corrections** (*Boolean*) – If `True`, use regional corrections (preferentially only when using "new" Reese parameters).

Returns
m_out – Horizontal distribution of the melt rate in m ice/s for the ice shelf of interest.

Return type
xr.DataArray

multimelt.melt_functions.**calculate_melt_rate_2D_picop_1isf**(*kisf*, *T_S_profile*, *geometry_info_2D*, *geometry_info_1D*, *box_charac_all_2D*, *box_charac_all_1D*, *isf_stack_mask*, *mparam*, *box_tot_nb*, *gamma_pico*, *gamma_plume*, *C*, *E0*, *angle_option*, *pism_version*, *picop_opt*)

Function to compute melt from box parameterisations for one ice shelf.

Parameters

- **kisf** (*int*) – Ice shelf ID for the ice shelf of interest.
- **T_S_profile** (*xarray.Dataset*) – Dataset containing temperature (in degrees C) and salinity (in psu) input profiles.
- **geometry_info_2D** (*xarray.Dataset*) – Dataset containing relevant 2D geometrical information.
- **geometry_info_1D** (*xarray.Dataset*) – Dataset containing relevant 1D geometrical information.
- **box_charac_all_2D** (*xarray.Dataset*) – Dataset containing relevant 2D box characteristics for all ice shelves.
- **box_charac_all_1D** (*xarray.Dataset*) – Dataset containing relevant 1D box characteristics for all ice shelves.
- **isf_stack_mask** (*xarray.DataArray*) – DataArray containing the stacked coordinates of the ice shelves (to make computing faster).
- **mparam** (*str*) – Parameterisation to be applied.
- **box_tot_nb** (*int*) – Total number of boxes
- **gamma_pico** (*float*) – Gamma to be used in PICO part.
- **gamma_plume** (*float*) – Gamma to be used in plume part.
- **C** (*float*) – Circulation parameter C (Sv m3 kg-1 = m6 kg-1 s-1) in [0.1;9]*1.e6.
- **E0** (*float*) – Entrainment coefficient.
- **angle_option** (*str*) – Slope to be used, choice between "cavity" (cavity slope), "lazero" (lazeroms18), "local" (local slope)
- **pism_version** (*str*) – Can be `yes` or `no`, depending on if you want to use the PICO version as implemented in PISM or the original PICO (uniform box melt). See Sec. 2.4 by Reese et al. 2018 for more info.
- **picop_opt** (*str*) – Can be `2019`, `2018` , depending on if you want to use PICOP with analytical plume param or with empirical plume param.

Returns

m_out – Horizontal distribution of the melt rate in m ice/s for the ice shelf of interest.

Return type

xr.DataArray

multimelt.melt_functions.**calculate_melt_rate_2D_plumes_1isf**(*kisf*, *T_S_profile*, *geometry_info_2D*, *geometry_info_1D*, *isf_stack_mask*, *mparam*, *gamma*, *E0*)

Function to compute melt from plume parameterisations for one ice shelf. 'lazero19': uses average of hydrographic properties extrapolated to local ice draft depth as ambient temperature and salinity 'lazero19_modif': Modification presented in Burgard et al. 2022

Parameters

- **kisf** (*int*) – Ice shelf ID for the ice shelf of interest.
- **T_S_profile** (*xarray.Dataset*) – Dataset containing temperature (in degrees C) and salinity (in psu) input profiles.
- **geometry_info_2D** (*xarray.Dataset*) – Dataset containing relevant 2D geometrical information.
- **geometry_info_1D** (*xarray.Dataset*) – Dataset containing relevant 1D geometrical information.
- **isf_stack_mask** (*xarray.DataArray*) – DataArray containing the stacked coordinates of the ice shelves (to make computing faster).
- **mparam** (*str*) – Parameterisation to be applied.
- **gamma** (*float*) – Gamma to be tuned.
- **E0** (*float*) – Entrainment coefficient.

Returns

melt_rate – Horizontal distribution of the melt rate in m/s for the ice shelf of interest.

Return type

xr.DataArray

multimelt.melt_functions.**calculate_melt_rate_2D_simple_1isf**(*kisf*, *T_S_profile*, *geometry_info_2D*, *geometry_info_1D*, *isf_stack_mask*, *mparam*, *gamma*, *U_param=True*, *HUB=False*)

Function to compute melt from simple parameterisations for one ice shelf.

Parameters

- **kisf** (*int*) – Ice shelf ID for the ice shelf of interest.
- **T_S_profile** (*xarray.Dataset*) – Dataset containing temperature (in degrees C) and salinity (in psu) input profiles.
- **geometry_info_2D** (*xarray.Dataset*) – Dataset containing relevant 2D geometrical information.
- **geometry_info_1D** (*xarray.Dataset*) – Dataset containing relevant 1D geometrical information.
- **isf_stack_mask** (*xarray.DataArray*) – DataArray containing the stacked coordinates of the ice shelves (to make computing faster).
- **mparam** (*str*) – Parameterisation to be applied.
- **gamma** (*float*) – Gamma to be tuned.
- **U_param** (*Boolean*) – If `True` we use the complex parameterisation of U, if `False`, this is "only" equal to (rho_sw * c_po) / (rho_i * L_i).

Returns

melt_rate – Horizontal distribution of the melt rate in m/s for the ice shelf of interest.

Return type
xr.DataArray

multimelt.melt_functions.**calculate_melt_rate_Gt_and_box1_all_isf**(*nisf_list*, *T_S_profile*, *geometry_info_2D*, *geometry_info_1D*, *isf_stack_mask*, *mparam*, *gamma*, *U_param=True*, *C=None*, *E0=None*, *angle_option='lazero'*, *box_charac_2D=None*, *box_charac_1D=None*, *box_tot=None*, *box_tot_option='box_nb_tot'*, *pism_version='no'*, *picop_opt='no'*, *gamma_plume=None*, *T_corrections=False*, *tuning_mode=False*, *HUB=False*, *verbose=True*)

Function to process input information and call the 2D and 1D functions to compute melt rate variables.

Parameters

- **nisf_list** (*array of int*) – List containing the ice shelf IDs for all ice shelves of interest.
- **T_S_profile** (*xarray.Dataset*) – Dataset containing temperature (in degrees C) and salinity (in psu) input profiles.
- **geometry_info_2D** (*xarray.Dataset*) – Dataset containing relevant 2D geometrical information.
- **geometry_info_1D** (*xarray.Dataset*) – Dataset containing relevant 1D geometrical information.
- **isf_stack_mask** (*xarray.DataArray*) – DataArray containing the stacked coordinates of the ice shelves (to make computing faster).
- **mparam** (*str*) – Parameterisation to be applied.
- **gamma** (*float*) – Gamma to be tuned.
- **U_param** (*Boolean*) – If `True` we use the complex parameterisation of U, if `False`, this is "only" equal to (rho_sw * c_po) / (rho_i * L_i). Relevant for simple parameterisations only.
- **C** (*float*) – Circulation parameter C (Sv m3 kg-1 = m6 kg-1 s-1) in [0.1;9]*1.e6.
- **E0** (*float*) – Entrainment coefficient.
- **angle_option** (*str*) – Slope to be used, choice between "cavity" (cavity), "lazero" (lazeroms18), "local" (local)
- **box_charac_2D** (*xarray.Dataset*) – Dataset containing relevant 2D box characteristics for all ice shelves.
- **box_charac_1D** (*xarray.Dataset*) – Dataset containing relevant 1D box characteristics for all ice shelves.
- **box_tot** (*int*) – Either the total number of boxes being used if box_tot_option='box_nb_tot' or the configuration to use if box_tot_option='nD_config'.

- **box_tot_option** (*str*) – Defines how box_tot``should be interpreted. Can be either 'box_nb_tot', then ``box_tot is the total number of boxes or 'nD_config', then box_tot is the configuration to use.
- **pism_version** (*str*) – Can be yes or no, depending on if you want to use the PICO version as implemented in PISM or the original PICO (uniform box melt). See Sec. 2.4 by Reese et al. 2018 for more info.
- **picop_opt** (*str*) – Can be yes or no, depending on if you want to use PICOP or the original PICO.
- **gamma_plume** (*float*) – Gamma to be used in the plume part of PICOP.
- **T_corrections** (*Boolean*) – If True, use regional corrections (preferentially only when using "new" Reese parameters).
- **tuning_mode** (*Boolean*) – If True, only compute integrated melt.
- **verbose** (*Boolean*) – True if you want the program to keep you posted on where it is in the calculation.

Returns
out_1D – Containing the melt in Gt/yr for each ice shelf and the mean melt rate in m/yr i

Return type
xarray.Dataset

multimelt.melt_functions.**compute_M_hat**(*X_hat*)

Compute M_hat for plume parameterisation.

This function computes the M_hat (or M0) for the plume parameterisation. This is Equation 26 in Lazeroms et al. 2019.

Parameters
X_hat (*scalar or array*) – Coordinate describing distance from plume origin.

Returns
M_hat – Dimensionless melt rate, to be multiplied with the Mterm in Eq 28a.

Return type
scalar or array

multimelt.melt_functions.**compute_M_hat_picop**(*x_hat*)

Compute Mhat factor for PICOP plume parameterisation.

This function computes the Mhat factor for the Lazeroms 2018 plume parameterisation. This is Equation A13 in Lazeroms et al. 2018.

Parameters
x_hat (*array*) – Factor to be used for the polynomial version of M_hat. Dimensionless coordinate.

Returns
M_hat_picop – Factor to be multiplied with Mterm.

Return type
array

multimelt.melt_functions.**compute_Mterm**(*T_in*, *S_in*, *Tf*, *c_rho_1*, *c_tau*, *gamma*, *E0*, *alpha*, *thermal_forcing_factor*)

Compute M-term for plume parameterisation.

This function computes the M-term for the plume parameterisation. This is the beginning of Equation 28(a) in Lazeroms et al. 2019.

Parameters

- **T_in** (*scalar (or array?)*) – Ambient temperature in degrees C.
- **S_in** (*scalar (or array?)*) – Ambient salinity in psu.
- **Tf** (*scalar (or array?)*) – Freezing temperature
- **c_rho_1** (*scalar*) – Constant given in Table 1 in Lazeroms et al. 2019.
- **c_tau** (*scalar (or array?)*) – Constant given in Table 1 in Lazeroms et al. 2019.
- **gamma** (*scalar*) – Effective thermal Stanton number. Can be modulated for tuning.
- **E0** (*scalar*) – Entrainment coefficient. Can be modulated for tuning.
- **alpha** (*scalar or array*) – Slope angle in rad (must be positive).
- **thermal_forcing_factor** (*scalar (or array?)*) – Factor to be multiplied to T0-Tf in the end of Eq. 28a - either thermal forcing or thermal forcing average.

Returns
Mterm – Term to be multiplied with M_hat in Eq 28a.

Return type
scalar or array

multimelt.melt_functions.**compute_Mterm_picop**(*T_in*, *Tf*, *E0*, *alpha*, *gamma_T_S*)

Compute M-term for PICOP plume parameterisation.

This function computes the M-term for the PICOP plume parameterisation. This is Equation 10 in Pelle et al. 2019.

Parameters

- **T_in** (*scalar (or array?)*) – Ambient temperature in degrees C.
- **Tf** (*scalar (or array?)*) – Freezing temperature
- **E0** (*scalar*) – Entrainment coefficient. Can be modulated for tuning.
- **alpha** (*scalar or array*) – Slope angle in rad (must be positive).
- **gamma_T_S** (*float*) – Effective Stanton number.

Returns
Mterm – Term to be multiplied with M_hat in Eq 9 of Pelle et al. (2019).

Return type
scalar or array

multimelt.melt_functions.**compute_T_S_one_box_PICO**(*T_prev_box*, *S_prev_box*, *box_depth*, *box_area*, *box_nb*, *C*, *gamma*, *q*)

Compute T and S of a box in PICO.

This function computes the temperature and salinity of a box in the box model PICO (see Reese et al., 2018).

Parameters

- **T_prev_box** (*scalar*) – Temperature in the previous box (box n-1) in degrees C.
- **S_prev_box** (*scalar*) – Salinity in the previous box (box n-1) in psu.
- **box_depth** (*scalar*) – Depth of the box below the ice in m (depth is negative!).

- **box_area** (*scalar*) – Area of the box in m^2.
- **box_nb** (*scalar*) – Number of the current box (n).
- **C** (*scalar*) – Circulation parameter C in Sv*:math:*mathtt{m}^{3}* `*:math:`mathtt{kg}^{-1}` = $m^6$`$*:math:$mathtt{kg}^{-1}*:math:`mathtt{s}^{-1}) in [0.1;9]*1.e6
- **gamma** (*scalar*) – Effective turbulent temperature exchange velocity in m per second
- **q** (*scalar*) – Overturning flux q in m^3 per second

Returns

- **q** (*scalar*) – Overturning flux q in m^3 per second
- **T_cur_box** (*scalar*) – Temperature in the current box (box n) in degrees C.
- **S_cur_box** (*scalar*) – Salinity in the current box (box n) in psu.

multimelt.melt_functions.**compute_X_hat**(*ice_draft_depth*, *zGL*, *T_in*, *Tf*, *E0*, *c_tau*, *alpha*, *gamma*)

Compute x_hat (or x_tilda) for plume parameterisation.

This function computes x_hat (or x_tilda) for the plume parameterisation. It is a dimensionless coordinate describing distance from plume origin. This is Equation 28(b) in Lazeroms et al. 2019.

Parameters

- **ice_draft_depth** (*scalar (or array?)*) – Depth of the ice draft in m (depth is negative!).
- **zGL** (*scalar (or array?)*) – Depth of the grounding line where the source of the plume is in m (depth is negative!).
- **T_in** (*scalar (or array?)*) – Ambient temperature in degrees C.
- **Tf** (*scalar (or array?)*) – Freezing temperature
- **E0** (*scalar*) – Entrainment coefficient. Can be modulated for tuning.
- **c_tau** (*scalar (or array?)*) – Constant given in Table 1 in Lazeroms et al. 2019.
- **alpha** (*scalar or array*) – Slope angle in rad (must be positive).
- **gamma** (*scalar*) – Effective thermal Stanton number. Can be modulated for tuning.

Returns

x_hat – Dimensionless coordinate describing distance from plume origin. Has to be between 0 and 1

Return type

scalar or array

multimelt.melt_functions.**compute_X_hat_picop**(*T_in*, *Tf*, *ice_draft_depth*, *zGL*, *stanton_number*, *E0*, *alpha*)

Compute xhat factor for PICOP plume parameterisation.

This function computes the x-hat factor for the plume parameterisation. This is Equation 8 in Pelle et al. 2019.

Parameters

- **T_in** (*scalar (or array?)*) – Ambient temperature in degrees C.
- **Tf** (*scalar (or array?)*) – Freezing temperature
- **ice_draft_depth** (*scalar (or array?)*) – Depth of the ice draft in m. Must be negative.

- **zGL** (*scalar (or array?)*) – Depth of the grounding line in m. Must be negative.
- **stanton_number** (*float*) – Effective Stanton Number.
- **E0** (*scalar*) – Entrainment coefficient. Can be modulated for tuning.
- **alpha** (*scalar or array*) – Slope angle in rad (must be positive).

Returns
x_hat – Factor to be used for the polynomial version of M_hat. Dimensionless coordinate.

Return type
array

multimelt.melt_functions.**compute_c_rho_tau**(*gamma*, *S_in*)

Compute the c-constants for plume parameterisation.

This function computes c_rho_1, c_rho_2 and c_tau for the plume parameterisation. They are constants given in Table 1 of Lazeroms et al. 2019.

Parameters

- **gamma** (*scalar*) – Effective thermal Stanton number. Can be modulated for tuning.
- **S_in** (*scalar (or array?)*) – Ambient salinity in psu.

Returns

- **c_rho_1** (*scalar (or array?)*) – Constant given in Table 1 in Lazeroms et al. 2019.
- **c_rho_2** (*scalar*) – Constant given in Table 1 in Lazeroms et al. 2019.
- **c_tau** (*scalar (or array?)*) – Constant given in Table 1 in Lazeroms et al. 2019.

multimelt.melt_functions.**convert_kg_ice_per_m2_to_Gt**(*melt_rate_kg_ice_per_m2*, *grid_cell_area*)

Convert kg per m2 ice in Gt.

This function converts an ice kg per m2 into a mass in Gt.

Parameters

- **melt_rate_kg_ice_per_m2** (*scalar or array*) – Quantity in kg of ice per m^2 (e.g. melt rate)
- **grid_cell_area** (*scalar or array*) – Area of the grid cells in m^2. if array: same shape as melt_rate_m_ice.

Returns
melt_rate_m_Gt – Quantity in Gt (e.g. melt rate)

Return type
scalar or array

multimelt.melt_functions.**convert_m_ice_to_Gt**(*melt_rate_m_ice*, *grid_cell_area*)

Convert m ice in Gt.

This function converts an ice thickness into a mass in Gt.

Parameters

- **melt_rate_m_ice** (*scalar or array*) – Quantity in m of ice (e.g. melt rate)
- **grid_cell_area** (*scalar or array*) – Area of the grid cells in m^2. if array: same shape as melt_rate_m_ice.

Returns
melt_rate_m_Gt – Quantity in Gt (e.g. melt rate)

Return type
scalar or array

multimelt.melt_functions.**convert_m_ice_to_m_water**(*melt_rate_m_ice*, *grid_cell_area*)

Convert m ice in m water equivalent.

This function converts an ice thickness into a water equivalent height.

Parameters

- **melt_rate_m_ice** (*scalar or array*) – Quantity in m of ice (e.g. melt rate)
- **grid_cell_area** (*scalar or array*) – Area of the grid cells in m^2. if array: same shape as melt_rate_m_ice.

Returns
melt_rate_m_w – Quantity in m of water equivalent (e.g. melt rate)

Return type
scalar or array

multimelt.melt_functions.**f_xterm**(*stanton_number*, *E0*, *alpha*)

Compute f-term for x-term in PICOP for plume parameterisation.

This function computes the f-term for x-term in PICOP. This is the right part of Equation A10 in Lazeroms et al. 2018.

Parameters

- **stanton_number** (*float*) – Effective Stanton number computed based on Equation 5 in Pelle et al. 2019.
- **E0** (*float*) – Entrainment coefficient.
- **alpha** (*scalar or array*) – Slope angle in rad (must be positive).

Returns
f_alpha – Term to be used for the length scale l in Equation A10 in Lazeroms et al. 2018.

Return type
scalar or array

multimelt.melt_functions.**find_closest_depth_levels**(*depth_levels*, *depth_of_int*)

Find closest depth levels in an array to a given depth.

This function looks for nearest neighbors of a given depth in an array of depth levels. Depth must be positive!

Parameters

- **depth_levels** (*xarray.DataArray*) – DataArray cointaining the dimension 'depth' with the depth levels you want to localise your depth in. Depth must be positive!
- **depth_of_int** (*xarray.DataArray*) – Depth of interest (can be more than 1D). Depth must be positive!

Returns

- **lev_inf** (*xarray.DataArray*) – nearest depth level below depth of interest
- **lev_sup** (*xarray.DataArray*) – nearest depth level above depth of interest
- **weight_inf** (*xarray.DataArray*) – weight to nearest depth level below depth of interest
- **weight_sup** (*xarray.DataArray*) – weight to nearest depth level above depth of interest

multimelt.melt_functions.**freezing_temperature**(*salinity*, *depth*)

Compute freezing temperature.

This function computes the melting-freezing point Tf at the interface between the ocean and the ice-shelf basal surface. Formula and coefficients taken from Favier et al., 2019. Depth must be negative below sea level!

Parameters

- **salinity** (*scalar or array*) – Practical salinity in psu
- **depth** (*scalar or array*) – Depth in m, must be negative below sea level

Returns
T_f – Melting-freezing point Tf at the interface between the ocean and the ice-shelf basal surface

Return type
scalar or array

multimelt.melt_functions.**g_Mterm**(*alpha*, *E0*, *gamma_T_S*)

Compute g-term for M-term in PICOP for plume parameterisation.

This function computes the g-term for M-term in PICOP. This is the beginning of Equation 6 in Pelle et al. 2019.

Parameters

- **alpha** (*scalar or array*) – Slope angle in rad (must be positive).
- **E0** (*float*) – Entrainment coefficient.
- **gamma_T_S** (*float*) – Effective Stanton number.

Returns
g_alpha – Term used to compute the melt in Equation 10 of Pelle et al. 2019.

Return type
scalar or array

multimelt.melt_functions.**interp_from_levels_to_depth_of_int**(*var*, *lev_inf*, *lev_sup*, *weight_inf*, *weight_sup*)

Interpolate a variable from the nearest neighbours temperature.

This function computes from a variable weighted from the nearest neighbours of a given depth in an array of depth levels. Depth must be positive!

Parameters

- **var** (*xarray.DataArray*) – DataArray of the variable of interest cointaining the dimension 'depth'. Depth must be positive!
- **lev_inf** (*xarray.DataArray*) – nearest depth level below depth of interest
- **lev_sup** (*xarray.DataArray*) – nearest depth level above depth of interest
- **weight_inf** (*xarray.DataArray*) – weight to nearest depth level below depth of interest
- **weight_sup** (*xarray.DataArray*) – weight to nearest depth level above depth of interest

Returns
new_var – interpolated value

Return type
xarray.DataArray

multimelt.melt_functions.**interp_profile_to_depth_of_interest**(*var_in*, *depth_of_int*)

Interpolate a profile to a given depth point.

This function interpolates a variable at a given depth from a profile.

Parameters

- **var_in** (*xr.DataArray*) – Profile of the variable of interest, must have a dimension 'depth'. Make sure that depth_of_int and the dimension depth have the same sign.
- **depth_of_int** (*scalar*) – Depth at which we want the variable interpolated.

Returns
var_out – Variable of interest at the given depth of interest.

Return type
xr.DataArray

multimelt.melt_functions.**linear_local_param**(*gamma*, *melt_factor*, *thermal_forcing*)

Apply the linear local parametrization.

This function computes the basal melt based on a linear local parametrization (see Favier et al., 2019 or Beckmann and Goosse, 2003).

Parameters

- **gamma** (*scalar*) – Heat exchange velocity in m per second
- **melt_factor** (*scalar*) – Melt factor representing (rho_sw*c_pw) / (rho_i*L_i) in K^{-1}.
- **thermal_forcing** (*scalar or array*) – Difference between T and the freezing temperature Tf (T-Tf) in K or degrees C

Returns
melt – Melt rate in m ice per second

Return type
scalar or array

multimelt.melt_functions.**merge_over_dim**(*da_in*, *da_out*, *dim*, *dim_index*)

Utility function to merge different melt rate results into one DataArray.

This function merges different melt rate results into one DataArray (e.g. different param choices or different ice shelves).

Parameters

- **da_in** (*xr.DataArray*) – DataArray to be merged to the rest.
- **da_out** (*xr.DataArray*) – DataArray merged until now.
- **dim** (*str*) – Name of the dimension
- **dim_index** (*str or int*) – Index of dim corresponding to da_in.

Returns
da_out – Merged DataArray.

Return type
xr.DataArray

multimelt.melt_functions.**plume_param**(*T_in*, *S_in*, *ice_draft_depth*, *zGL*, *alpha*, *gamma*, *E0*, *picop=False*)

Apply the plume parametrization.

This function computes the basal melt based on a plume parametrization (see Lazeroms et al. 2018 and Lazeroms et al. 2019).

Parameters

- **T_in** (*scalar (or array?)*) – Ambient temperature in degrees C.
- **S_in** (*scalar (or array?)*) – Ambient salinity in psu.
- **ice_draft_depth** (*scalar or array*) – Depth of the ice draft in m (depth is negative!).
- **zGL** (*scalar or array*) – Depth of the grounding line where the source of the plume is in m (depth is negative!).
- **alpha** (*scalar or array*) – Slope angle in rad (must be positive).
- **gamma** (*scalar*) – Effective thermal Stanton number. Can be modulated for tuning.
- **E0** (*scalar*) – Entrainment coefficient. Can be modulated for tuning.
- **picop** (*Boolean*) – Option defining which Mterm function to use.

Returns
melt_rate – Melt rate in m ice per second.

Return type
scalar or array

multimelt.melt_functions.**plume_param_modif**(*theta_isf*, *salinity_isf*, *ice_draft_points*, *front_bot_dep_max_isf*, *zGL_isf*, *alpha_isf*, *conc_isf*, *dGL_isf*, *gamma*, *E0*)

Apply the plume parametrization with modifications (from Burgard et al. 2022).

This function computes the basal melt based on a plume parameterisation (see Lazeroms et al. 2018 and Lazeroms et al. 2019) with modifications from pers. comm. between A. Jenkins and C. Burgard & N. Jourdain.

Parameters

- **theta_isf** (*array*) – Ambient temperature profile in degrees C.
- **salinity_isf** (*array*) – Ambient salinity profile in psu.
- **ice_draft_points** (*array*) – Depth of the ice draft in m (depth is positive!).
- **front_bot_dep_max_isf** (*scalar*) – Maximum depth of the continental shelf at the entry of the ice shelf (depth is positive!)
- **zGL_isf** (*scalar or array*) – Depth of the grounding line in m (depth is negative!), containing 'cavity' and 'lazero' option.
- **alpha_isf** (*scalar or array*) – Slope angle in rad (must be positive), containing 'cavity' and 'lazero' option.
- **conc_isf** (*array*) – Concentration of grid cell covered by the ice shelf.
- **dGL_isf** (*array*) – Distance between local point and nearest grounding line in m.
- **Tf_gl** (*scalar (or array?)*) – Freezing temperature at the grounding line in degrees C.
- **E0** (*scalar*) – Entrainment coefficient. Can be modulated for tuning.
- **gamma** (*scalar*) – Effective thermal Stanton number. Can be modulated for tuning.

Returns
melt_rate – Melt rate in m ice per second.

Return type
scalar or array

multimelt.melt_functions.**quadratic_local_param**(*gamma*, *melt_factor*, *thermal_forcing*, *U_factor*)

Apply the quadratic local parametrization.

This function computes the basal melt based on a quadratic local parametrization (based on Favier et al., 2019 or DeConto and Pollard, 2016), revisited in Burgard et al. 2021.

Parameters

- **gamma** (*scalar*) – Heat exchange velocity in m per second
- **melt_factor** (*scalar*) – Melt factor representing (rho_sw*c_pw) / (rho_i*L_i) in K^{-1}.
- **thermal_forcing** (*scalar or array*) – Difference between T and the freezing temperature Tf (T-Tf) in K or degrees C
- **U_factor** (*scalar or array*) – Factor introduced to emulate the speed of the current, see function calculate_melt_rate_2D_simple_1isf.

Returns
melt – Melt rate in m ice per second

Return type
scalar or array

multimelt.melt_functions.**quadratic_mixed_mean**(*gamma*, *melt_factor*, *thermal_forcing*, *thermal_forcing_avg*, *U_factor*)

Apply the quadratic local and non-local parametrization.

This function computes the basal melt based on a quadratic local parametrization (see Favier et al., 2019), revisited in Burgard et al. 2021.

Parameters

- **gamma** (*scalar*) – Heat exchange velocity in m per second
- **melt_factor** (*scalar*) – Melt factor representing (rho_sw*c_pw) / (rho_i*L_i) in K^{-1}.
- **thermal_forcing** (*scalar or array*) – Difference between T and the freezing temperature Tf (T-Tf) in K or degrees C
- **thermal_forcing_avg** (*scalar*) – Spatial average of the thermal forcing in K or degrees C
- **U_factor** (*scalar or array*) – Factor introduced to emulate the speed of the current, see function calculate_melt_rate_2D_simple_1isf.

Returns
melt – Melt rate in m ice per second

Return type
scalar or array

multimelt.melt_functions.**quadratic_mixed_slope**(*gamma*, *melt_factor*, *thermal_forcing*, *thermal_forcing_avg*, *U_factor*, *alpha*)

Apply the quadratic local and non-local parametrization taking into account the slope.

This function computes the basal melt based on a quadratic local parametrization and taking into account the slope (see Jourdain et al. 2020), revisited in Burgard et al. 2021.

Parameters

- **gamma** (*scalar*) – Heat exchange velocity in m per second
- **melt_factor** (*scalar*) – Melt factor representing (rho_sw*c_pw) / (rho_i*L_i) in K^{-1}.

- **thermal_forcing** (*scalar or array*) – Difference between T and the freezing temperature Tf (T-Tf) in K or degrees C
- **thermal_forcing_avg** (*scalar*) – Spatial average of the thermal forcing in K or degrees C
- **U_factor** (*scalar or array*) – Factor introduced to emulate the speed of the current, see function calculate_melt_rate_2D_simple_1isf.
- **alpha** (*scalar or array*) – Slope angle in rad (must be positive).

Returns
melt – Melt rate in m ice per second

Return type
scalar or array

1.6.7 plume_functions

multimelt.plume_functions.**add_GL_max**(*plume_var_of_int*, *ice_draft_pos*)

Add the information about the deepest grounding line point to the info-dataset.

Parameters

- **plume_var_of_int** (*xr.Dataset*) – Dataset containing at least `'GL_mask'` and `'dGL_dIF'`.
- **ice_draft_pos** (*xr.DataArray*) – Ice draft depth in m. Positive downwards.

Returns
plume_var_of_int – Dataset extended with `'GL_max'` and `'dGL_max_dIF'`.

Return type
xr.Dataset

multimelt.plume_functions.**check_slope_one_dimension**(*input_da*, *shifted_plus*, *shifted_minus*, *dx*)

Compute the basal slope at each point.

Parameters

- **input_da** (*xr.DataArray*) – Array where slope needs to be checked. For example: ice draft.
- **shifted_plus** (*xr.DataArray*) – Shifted version (positive direction) of input_da.
- **shifted_minus** (*xr.DataArray*) – Shifted version (negative direction) of input_da.
- **dx** (*float*) – Step in the coordinate along which input_da was shifted

Returns
slope – slope along that coordinate, is 0 if nan

Return type
xr.DataArray

multimelt.plume_functions.**compute_alpha_cavity**(*plume_var_of_int*)

Compute alpha with a very simple approach (angle between horizontal and ice front) => cavity slope

Parameters
plume_var_of_int (*xr.DataArray or xr.Dataset*) – Dataset containing `'GL_max'`, `'front_ice_depth_avg'` and `'dGL_max_dIF'`

Returns
alphas – Angle at origin of plume in rad.

Return type
xr.DataArray

multimelt.plume_functions.**compute_alpha_local**(*kisf*, *plume_var_of_int*, *ice_draft_neg*, *dx*, *dy*)

Compute alphas like in Appendix B of Favier et al., 2019 TCDiscussions.

Parameters

- **kisf** (*int*) – ID of the ice shelf of interest
- **plume_var_of_int** (*xr.Dataset*) – Dataset containing `'ISF_mask'` and `'dIF'`
- **ice_draft_neg** (*xr.DataArray*) – Ice draft depth in m. Negative downwards.
- **dx** (*float*) – Grid spacing in the x-direction
- **dy** (*float*) – Grid spacing in the y-direction

Returns
go_back_to_whole_grid_local_alpha – Local slope angle in rad for each point.

Return type
xr.DataArray

multimelt.plume_functions.**compute_zGL_alpha_all**(*plume_var_of_int*, *opt*, *ice_draft_neg*)

Compute grounding line and angle for the plume for all ice shelves.

Parameters

- **plume_var_of_int** (*xr.Dataset*) – Dataset containing 'ISF_mask', 'GL_mask', 'IF_mask', 'dIF', 'dGL_dIF', 'latitude', 'longitude', 'front_ice_depth_avg'
- **opt** (*str*) –

  Method after which to compute the depth and angle. Can be:
  cavity : Zgl and Alpha are found between draft point and deepest GL point. lazero: original from Lazeroms et al. 2018 local: local slope
- **ice_draft_neg** (*xr.DataArray*) – Ice draft depth in m. Negative downwards.

Returns

- **plume_alpha** (*xr.DataArray*) – Angle in rad for each point.
- **plume_zGL** (*xr.DataArray*) – Depth of the grounding line in m at the plume origin for each point (negative downwards).

multimelt.plume_functions.**compute_zGL_alpha_lazero**(*kisf*, *plume_var_of_int*, *ice_draft_neg*, *dx*, *dy*)

Compute zGL and alphas in the Lazeroms approach.

This function computes zGl and alphas following the methodology in Lazeroms et al., 2018.

Parameters

- **kisf** (*int*) – ID of the ice shelf of interest
- **plume_var_of_int** (*xr.Dataset*) – Dataset containing `'ISF_mask'` and `'GL_mask'`
- **ice_draft_neg** (*xr.DataArray*) – Ice draft depth in m. Negative downwards.
- **ds_isf_lazeroms** (*xr.Dataset*) – Dataset containing the grounding line depth at the origin of the plume (`'gl_depth'`) and the gradient between the ice draft depth and the grounding line (`'gl_gradient'`)

- **dx** (*float*) – Grid spacing in the x-direction
- **dy** (*float*) – Grid spacing in the y-direction

Returns

- **alpha** (*xr.DataArray*) – Mean angle at the plume origin in rad for each point.
- **zGL** (*xr.DataArray*) – Mean depth of the grounding line in m at the plume origin for each point (negative downwards).

multimelt.plume_functions.**create_16_dir_weights**()

Prepare correlation filter in 16 directions.

This function prepares the correlation filter in 16 directions, following the methodology in Lazeroms et al;, 2018.

Returns
ds_weights – Weights for the filter and information about the x- and y-shift in the 16 directions.

Return type
xr.Dataset

multimelt.plume_functions.**first_criterion_lazero**(*kisf*, *plume_var_of_int*, *ice_draft_neg_isf*, *isf_and_GL_mask*, *ds_weights*, *dx*, *dy*)

Define first criterion for the plume parameters.

This function computes the basal slope and identifies the first criterion, following the methodology in Lazeroms et al;, 2018.

Parameters

- **kisf** (*int*) – ID of the ice shelf of interest
- **plume_var_of_int** (*xr.Dataset*) – Dataset containing `'ISF_mask'` and `'GL_mask'`
- **ice_draft_neg_isf** (*xr.DataArray*) – Ice draft depth for the given ice shelf in m. Negative downwards.
- **isf_and_GL_mask** (*xr.DataArray*) – Mask of the domain covered by the ice shelf and the grounding line (this extra mask is needed if the grounding line is defined on ground points)
- **ds_weights** (*xr.Dataset*) – Weights for the filter and information about the x- and y-shift in the 16 directions.
- **dx** (*float*) – Grid spacing in the x-direction
- **dy** (*float*) – Grid spacing in the y-direction

Returns

- **GL_depth** (*xr.DataArray*) – Depth of the grounding line points (negative downwards).
- **sn_isf** (*xr.DataArray*) – Basal slope in all 16 directions
- **first_crit** (*xr.DataArray*) – Boolean where sn_sf > 0
- *draft_depth* – Ice draft depth in m (negative downwards) extended through the 'direction' dimension.

multimelt.plume_functions.**nd_corr**(*input1*, *weights*)

Correlation using a filter (scipy.ndimage library).

Parameters

- **input1** (*xr.DataArray*) – Array to be filtered.
- **weights** (*xr.DataArray*) – Filter to be applied.

Return type
Filtered data (sum according to the weights)

multimelt.plume_functions.**nd_corr_sig**(*input1*, *weights*)

Correlation using a filter (scipy.signal library).

Parameters

- **input1** (*xr.DataArray*) – Array to be filtered.
- **weights** (*xr.DataArray*) – Filter to be applied.

Return type
Filtered data (sum according to the weights)

multimelt.plume_functions.**prepare_filter_16dir_isf**(*isf_and_GL_mask*, *GL_depth*, *x_size*, *y_size*, *ds_weights*)

Prepare the filter to check grounding line in all 16 directions.

This function computes the basal slope and identifies the first criterion, following the methodology in Lazeroms et al;, 2018.

Parameters

- **isf_and_GL_mask** (*xr.DataArray*) – Mask of the domain covered by the ice shelf and the grounding line (this extra mask is needed if the grounding line is defined on ground points)
- **GL_depth** (*xr.DataArray*) – Depth of the grounding line points (negative downwards).
- **x_size** (*int*) – Size of the domain in the x-coordinate.
- **y_size** (*int*) – Size of the domain in the y-coordinate
- **ds_weights** (*xr.Dataset*) – Weights for the filter and information about the x- and y-shift in the 16 directions.

Returns

- **ds_isf_weights** (*xr.Dataset*) – Weights for the filter and information about the x- and y-shift in the 16 directions adapted to the given ice shelf domain.
- **mid_coord** (*int*) – Maximum dimension of the domain and therefore middle of the filter.

multimelt.plume_functions.**prepare_plume_charac**(*plume_param_options*, *ice_draft_pos*, *plume_var_of_int*)

Overall function to compute the plume characteristics depending on geometry.

Parameters

- **plume_param_options** (*list of str*) – Parametrization options (typically 'cavity', 'lazero' and 'local').
- **ice_draft_pos** (*xr.DataArray*) – Ice draft depth in m. Positive downwards.
- **plume_var_of_int** (*xr.Dataset*) – Dataset containing 'ISF_mask', 'GL_mask', 'IF_mask', 'dIF', 'dGL_dIF', 'latitude', 'longitude', 'front_ice_depth_avg'

Returns
outfile – Dataset containing plume characteristics depending on geometry needed for plume parametrization.

Return type
xr.Dataset

multimelt.plume_functions.**prepare_plume_dataset**(*plume_var_of_int*, *plume_param_options*)

Prepare plume dataset to include zGL and alpha.

This initializes the grounding line depth and the angle at the origin of the plume.

Parameters

- **plume_var_of_int** (*xr.DataArray or xr.Dataset*) – Dataset or DataArray representing the domain.
- **plume_param_options** (*list of str*) – Parametrization options (typically 'cavity', 'lazero' and 'local').

Returns

plume_var_of_int – Dataset extended with `'alpha'` and `'zGL'`.

Return type

xr.Dataset

multimelt.plume_functions.**second_criterion_lazero**(*kisf*, *plume_var_of_int*, *ds_isf_weights*, *GL_depth*, *mid_coord*, *draft_depth*)

Define second criterion for the plume parameters.

This function looks for the grounding line depth at the plume origin and identifies the second criterion, following the methodology in Lazeroms et al., 2018.

Parameters

- **kisf** (*int*) – ID of the ice shelf of interest
- **plume_var_of_int** (*xr.Dataset*) – Dataset containing `'ISF_mask'`
- **ds_isf_weights** (*xr.Dataset*) – Weights for the filter and information about the x- and y-shift in the 16 directions adapted to the given ice shelf domain.
- **GL_depth** (*xr.DataArray*) – Depth of the grounding line points (negative downwards).
- **mid_coord** (*int*) – Maximum dimension of the domain and therefore middle of the filter.
- **draft_depth** – Ice draft depth in m (negative downwards) extended through the 'direction' dimension.

Returns

- **ds_isf_lazeroms** (*xr.Dataset*) – Dataset containing the grounding line depth at the origin of the plume (`'gl_depth'`) and the gradient between the ice draft depth and the grounding line (`'gl_gradient'`)
- **second_crit** (*xr.DataArray*) – Boolean where the gradient between the ice draft depth and the grounding line is negative.

multimelt.plume_functions.**summarize_alpha_zGL_lazero**(*kisf*, *ds_isf_lazeroms*, *sn_isf*, *first_crit*, *second_crit*, *plume_var_of_int*, *draft_depth*)

Summarize all criteria to compute zGL and alphas in the Lazeroms approach.

This function summarizes the first and second criteria to infer zGl and alphas following the methodology in Lazeroms et al., 2018.

Parameters

- **kisf** (*int*) – ID of the ice shelf of interest
- **ds_isf_lazeroms** (*xr.Dataset*) – Dataset containing the grounding line depth at the origin of the plume (`'gl_depth'`) and the gradient between the ice draft depth and the grounding line (`'gl_gradient'`)

- **sn_isf** (*xr.DataArray*) – Basal slope in all 16 directions
- **first_crit** (*xr.DataArray*) – Boolean where sn_sf > 0
- **second_crit** (*xr.DataArray*) – Boolean where the gradient between the ice draft depth and the grounding line is negative.
- **plume_var_of_int** (*xr.Dataset*) – Dataset containing `'ISF_mask'` and `'GL_mask'`
- **draft_depth** – Ice draft depth in m (negative downwards) extended through the 'direction' dimension.

Returns

- **go_back_to_whole_grid_alpha** (*xr.DataArray*) – Mean angle at the plume origin in rad for each point.
- **go_back_to_whole_grid_zgl** (*xr.DataArray*) – Mean depth of the grounding line in m at the plume origin for each point (negative downwards).

`multimelt.plume_functions.`**xr_nd_corr**(*data*, *weights*)

Correlation using a filter (scipy.ndimage library).

Parameters

- **data** (*xr.DataArray*) – Array to be filtered with at least dimensions ['y','x'].
- **weights** (*xr.DataArray*) – Filter to be applied with at least dimensions ['y0','x0'].

Return type
Filtered data (sum according to the weights)

`multimelt.plume_functions.`**xr_nd_corr_sig**(*data*, *weights*)

Correlation using a filter (scipy.signal library).

Parameters

- **data** (*xr.DataArray*) – Array to be filtered with at least dimensions ['y','x'].
- **weights** (*xr.DataArray*) – Filter to be applied with at least dimensions ['y0','x0'].

Return type
Filtered data (sum according to the weights)

`multimelt.plume_functions.`**xr_nd_corr_v2**(*data*, *weights*)

Correlation using a filter (scipy.ndimage library).

Parameters

- **data** (*xr.DataArray*) – Array to be filtered with at least dimensions ['y','x'].
- **weights** (*xr.DataArray*) – Filter to be applied with at least dimensions ['y0','x0'].

Return type
Filtered data (sum according to the weights)

1.6.8 useful_functions

multimelt.useful_functions.**bring_back_to_2D**(*stacked_da*)

multimelt.useful_functions.**change_coord_latlon_to_stereo**(*meshlon*, *meshlat*)

multimelt.useful_functions.**change_coord_stereo_to_latlon**(*meshx*, *meshy*)

multimelt.useful_functions.**choose_isf**(*var*, *isf_stacked_mask*, *kisf*)

multimelt.useful_functions.**create_stacked_mask**(*isfmask_2D*, *nisf_list*, *dims_to_stack*, *new_dim*)

multimelt.useful_functions.**cut_domain_stereo**(*var_to_cut*, *map_lim_x*, *map_lim_y*)

multimelt.useful_functions.**dist_sphere**(*lon1*, *lon2*, *lat1*, *lat2*)

multimelt.useful_functions.**in_range**(*in_xy*, *txy*)

multimelt.useful_functions.**weighted_mean**(*data*, *dims*, *weights*)

1.7 References

CHAPTER

TWO

HOW TO CITE MULTIMELT

The detailed description of the application of the functions in multimelt is found in Burgard et al., 2022 and should therefore, when used, be cited as follows:

Burgard, C., Jourdain, N. C., Reese, R., Jenkins, A., and Mathiot, P.: An assessment of basal melt parameterisations for Antarctic ice shelves, The Cryosphere Discuss. [preprint], https://doi.org/10.5194/tc-2022-32, in review, 2022.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[Beckmann & Goosse, 2003] Beckmann, A., & Goosse, H. (2003). A parameterization of ice shelfâocean interaction for climate models. *Ocean Modelling*, *5*(2), 157-170. doi:10.1016/S1463-5003(02)00019-7

[Burgard et al., 2022] Burgard, C., Jourdain, N.C., Reese, R., Jenkins, A., & Mathiot, P. (2022). An assessment of basal melt parameterisations for antarctic ice shelves. *The Cryosphere*, *16*, 4931â4975. doi:10.5194/tc-16-4931-2022

[DeConto & Pollard, 2016] DeConto, R.M., & Pollard, D. (2016). Contribution of antarctica to past and future sea-level rise. *Nature*, *531*(7596), 591-597. doi:10.1038/nature17145

[Favier et al., 2019] Favier, L., Jourdain, N.C., Jenkins, A., Merino, N., Durand, G., Gagliardini, O., … Mathiot, P. (2019). Assessment of sub-shelf melting parameterisations using the oceanâice-sheet coupled model nemo(v3.6)âelmer/ice(v8.3). *Geoscientific Model Development*, *12*(6), 2255-2283. doi:10.5194/gmd-12-2255-2019

[Holland et al., 2008] Holland, P.R., Jenkins, A., & Holland, D.M. (2008). The response of ice shelf basal melting to variations in ocean temperature. *Journal of Climate*, *21*(11), 2558-2572. doi:10.1175/2007JCLI1909.1

[Jenkins et al., 2018] Jenkins, A., Shoosmith, D., Dutrieux, P., Jacobs, S., Kim, T.W., Lee, S.H., … Stammerjohn, S. (2018). West antarctic ice sheet retreat in the amundsen sea driven by decadal oceanic variability. *Nature Geoscience*, *11*, 733â738. doi:10.1038/s41561-018-0207-4

[Jourdain et al., 2020] Jourdain, N.C., Asay-Davis, X., Hattermann, T., Straneo, F., Seroussi, H., Little, C.M., & Nowicki, S. (2020). A protocol for calculating basal melt rates in the ismip6 antarctic ice sheet projections. *The Cryosphere*, *14*(9), 3111-3134. doi:10.5194/tc-14-3111-2020

[Lazeroms et al., 2018] Lazeroms, W.M.J., Jenkins, A., Gudmunsson, G.H., & van de Wal, R.S.W. (2018). Modelling present-day basal melt rates for antarctic ice shelves using a parametrization of buoyant meltwater plumes. *The Cryosphere*, *12*(1), 49-70. doi:10.5194/tc-12-49-2018

[Lazeroms et al., 2019] Lazeroms, W.M.J., Jenkins, A., Rienstra, S.W., & van de Wal, R.S.W. (2019). An analytical derivation of ice-shelf basal melt based on the dynamics of meltwater plumes. *Journal of Physical Oceanography*, *49*(4), 917-939. doi:10.1175/JPO-D-18-0131.1

[Pelle et al., 2019] Pelle, T., Morlighem, M., & Bondzio, J.H. (2019). Brief communication: picop, a new ocean melt parameterization under ice shelves combining pico and a plume model. *The Cryosphere*, *13*(3), 1043-1049. doi:10.5194/tc-13-1043-2019

[Reese et al., 2018] Reese, R., Albrecht, T., Mengel, M., Asay-Davis, X., & Winkelmann, R. (2018). Antarctic sub-shelf melt rates via pico. *The Cryosphere*, *12*(6), 1969-1985. doi:10.5194/tc-12-1969-2018

[NEMO Team, 2019] NEMO Team. (2019). Nemo ocean engine. *Scientific Notes of Climate Modelling Center*, *27*. doi:10.5281/zenodo.1464816

PYTHON MODULE INDEX

m

INDEX

R

S

T

W

X