Supplement of Geosci. Model Dev., 16, 2607–2647, 2023
https://doi.org/10.5194/gmd-16-2607-2023-supplement

Geoscientific Model Development
Open Access
EGU

Supplement of

The CryoGrid community model (version 1.0) – a multi-physics toolbox for climate-driven simulations in the terrestrial cryosphere

Sebastian Westermann et al.

Correspondence to: Sebastian Westermann (sebastian.westermann@geo.uio.no)

S1: Quick start for CryoGrid community model

Software requirements: CryoGrid is written in Matlab, version 2018 or higher is required. For parallel applications, the Matlab parallel toolbox is required.

S1.1 Download and set up file structure

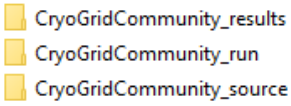
a) select or create a folder for the model code and results, e.g. “CryoGrid_git”

-download zip-file with CryoGrid source code, CryoGridCommunity_source.zip, from https://doi.org/10.5281/zenodo.6522424 and unpack in “CryoGrid_git”.

b) download the zip-file with run files to be modified by the user, “CryoGridCommunity_run.zip”, from https://doi.org/10.5281/zenodo.6522424 and unpack unpack in “CryoGrid_git”.

c) “CryoGrid_git” should now contain the subfolders “CryoGrid_git/CryoGridCommunity_source” and “CryoGrid_git/ CryoGridCommunity_run”, and “CryoGrid_git/CryoGridCommunity_source” the subfolders “source” and “UMLs”, while “CryoGrid_git/CryoGridCommunity_run” should contain the file “run_CG.m” (in addition to other files and folders).

d) Create a folder for parameter files and simulation results. For the default setting, create a folder “CryoGridCommunity_results” in “CryoGrid_git”, which should look like this:



S1.2 Set up parameter files

-using existing parameter files from S4 (recommended as first step)

download and unpack the file “CryoGridCommunity_parameter_files.zip” from https://doi.org/10.5281/zenodo.6522424. Copy the herein contained folders (i.e. “forcing”, “glacier”, etc.) in the folder “CryoGrid_git/CryoGridCommunity_results” (see S1.1 above). Each of the folders represents a simulation from Sect. 3.2, with folders named according to the run names (see Sl.4). The common model forcing is provided in the folder “forcing”.

-use automatic parameter file creator (recommended for advanced users)

a) edit and run the script “create_parameter_file_EXCEL.m” in “CryoGrid_git/ CryoGridCommunity_run”. A list of class names, their index and class options (for some classes) must be provided in the header of the file, as well as the folder in which he parameter files and results are stored (in this case “../CryoGridCommunity_results”). The script generates a parameter file

with default parameter values for each class, which the user must further edit manually. In the resulting parameter file, each parameter is associated with an explanatory comment, and it is possible to automatically enter a default value for many of the parameters.

S1.3 Run a CryoGrid simulation

a) edit the header of the script "run_CG.m" in "CryoGrid_git/CryoGridCommunity_run". When using a spreadsheet as parameter file (as for the parameter files provided in S4), set "init_format = 'EXCEL' and constant_file = 'CONSTANTS_excel'. Set the variable "run_name" to the name of the parameter file/ the folder in which it is stored (e.g. "reference_run" to simulate the reference run from Sect. 3.2) and the variable "result_path" to the folder in which the parameter files and results are stored (in this case "../CryoGridCommunity_results").

b) run the script "run_CG.m". Depending on the selected OUT class, results files in .mat-format will be written, typically as annual output files (as for the parameter files provided in S4). Note that for the parameter files provided in S4, no output files are written at the beginning of the simulation, i.e. during the accelerated spin-up. It generally requires a few hours of simulation time before the first output file is written.

S1.4 Quick look at selected simulation output

The script "read_output_and_display.m" provided in "CryoGrid_git/CryoGridCommunity_run" provides a simple possibility to display the depth- and time-resolved field of important model variables, such as temperature and volumetric contents of water and ice. The script must be run in sections: after the first section is run to set the internal paths to the model source code, the user must manually load the desired output mat-file (stored as annual time slices in the parameter files provided in S4). Then the rest of the script can be run, which results in five plots: temperature "T", volumetric water plus ice content "waterIce", volumetric water content "water" and volumetric ice content "ice", as well as the running number of the stratigraphy class used for a certain domain, with class 1 being the lowermost stratigraphy class. Users can adjust the displayed domain, ranging from a user-defined height above the ground surface (in order to display the seasonal snow cover), to a user-defined depth below the surface.

S2: Description of CryoGrid classes

S2.1 RUN_INFO

RUN_1D_ STANDARD: This is a "do-nothing" RUN_INFO class, which initializes and runs a single TILE class and has no other functionality. It basically serves as a "wrapper" around the TILE class and should be used for initial testing of applications, as well as for developing new stratigraphy classes.

RUN_1D_SPINUP: This class sequentially runs several TILE classes. It can be used for model spin-up, consecutively initializing the following TILE class with information on the ground thermal state computed by the previous TILE class (see below for options of TILE_BUILDER classes). RUN_1D_SPINUP can also be used to sequentially run several independent TILE classes, e.g. each for different input data sets or model parameters.

RUN_3D_STANDARD: Initializes and runs multi-tile simulations in a parallel computing environment, which can be used to run several independent TILE classes in parallel, e.g. each for different input data sets or model parameters.

S2.2 TILE, TILE_BUILDER and INIT_STEADY_STATE

TILE_1D_standard: initalizes and runs a CryoGrid stratigraphy. For initialization, different TILE_BUILDER classes can be used, which is a parameter in *TILE_1D_standard* and used to select the way the initial temperature profile is calculated. Depending on the choice of the TILE_BUILDER class, different sets of parameters must be defined by the user. The TILE_BUILDER *new_init* performs initialization using the information provided in the STRATIGRAPHY_STATVAR classes from the parameter files; *new_init_steady_state* computes a steady-state temperature profile taking the thermal conductivity of each grid cell, the lower boundary heat flux and a near-surface temperature at defined depth. These values can be either provided by the user, calculated with an INIT_STEADY_STATE class. *INIT_TTOP_from_forcing* applies the TTOP approach to the air temperatures provided by the model forcing, while *INIT_TTOP_from_out* uses the output of a previous TILE class stored by the OUT class *OUT_TDD_FDD*. This TILE_BUILDER class is used for the accelerated model spin-up procedure. The next TILE_BUILDER class, *update_forcing_out*, uses the last model state of the previous TILE (i.e. the stratigraphy will have the same stratigraphy classes), but overwrite the FORCING and OUT classes. This TILE_BUILDER class is used for the accelerated model spin-up procedure. *restart_OUT_last_timestep* reads a file written by the OUT class *OUT_last_timestep* (which saves the full model state either at defined time intervals or the very end) and continues the run from the point saved.

TILE_1D_standard2: same as *TILE_1D_standard*, but several OUT classes can be used instead of only a single OUT class, as in *TILE_1D_standard*. This is useful if one wants to not only store model output for the simulation period itself, but also the final state that can be used to restart a new TILE (with TILE_BUILDER *restart_OUT_last_timestep*, see above).

S2.3 FORCING

FORCING_seb: simple model forcing for stratigraphy classes computing the surface energy balance (keyword "*seb*"). The data must be stored in a Matlab ".mat" file with a struct *FORCING* with field "*data*", which contains the time series of the actual forcing data, e.g. *FORCING.data.Tair* contains the time series of air temperatures. Have a look at the existing forcing files in the folder "forcing" and prepare new forcing files in the same way. The mandatory forcing variables are air temperature (*Tair*, in °C), incoming long-wave radiation (*Lin*, in W/m2), incoming short-wave radiation (*Sin*, in W/m2), specific humidity (*q*, in kg water vapor / kg moist air), wind speed (*wind*, in m/sec), rainfall (*rainfall*, in mm/day), snowfall (*snowfall*, in mm/day) and timestamp (t_span, in Matlab time - increment 1 corresponds to one day). IMPORTANT POINT: the time series must be equally spaced in time, and this must be really exact. When reading the timestamps from an existing data set (e.g. an Excel file), rounding errors can result in small differences in the forcing timestep, often less than a second off. In this case, it is better to manually compile a new, equally spaced timestep in Matlab.

FORCING_seb _slope: same as FORCING_seb, but adjusts incoming short- and long-wave radiation to different aspects and slopes

S2.4 OUT

OUT_all: makes an identical copy of all stratigraphy classes at each output timestep, stored in a cell array in the variable STRATIGRAPHY. As the raw model state including parameters and temporary variables is stored, processing is required in order to analyze and display the output.

OUT_all _lateral: identical to *OUT_all*, but also stores the state variables of the lateral interaction classes (in the cell array LATERAL). This can in particular be used to obtain runoff curves if a lateral interaction classes for the water balance are used.

OUT_TDD_FDD: accumulates and stores depth profiles of thawing and freezing degree days over the entire simulation period. The resulting output file can e.g. be used by the INIT_STEADY_STATE class *INIT_TTOP_from_out* to initialize a steady-state temperature profile for a subsequent TILE class; used in the accelerated spin-up procedure.

OUT_last _timestep: stores the CryoGrid stratigraphy after defined time intervals, or the final state after the simulation has terminated. The output file is used to initialize a new TILE class with TILE_BUILDER *restart_OUT_last_timestep*, so that the simulation continues from the stored state. This is not only useful for troubleshooting model code, but also to perform ensemble simulations (e.g. for different future climate scenarios) starting from a common final state (e.g. from a historic simulation).

OUT_do_nothing: stores no output, generally used during model spin-up.

S2.5 GRID

GRID_user_defined: allows the user to define the model grid as layers with constant grid cell size, e.g. 0.05m in the uppermost two meters, then 0.1m to 5m depth, etc.

S2.6 STRATIGRAPHY_CLASSES

Note: the class type STRATIGRAPHY_CLASSES is not to be confused with stratigraphy classes, i.e. the classes that make up the CryoGrid stratigraphy (see below). Instead, it provides the information which stratigraphy classes are to be used for which depth layers, i.e. the "stratigraphy of stratigraphy classes".

STRAT_classes: provides information on the initial stratigraphy of stratigraphy classes, as well as other stratigraphy classes that are (potentially) needed during the run. The initial stratigraphy of stratigraphy classes is provided as a matrix with depth (i.e. the upper depth below the subsurface for each class - the last class extends to the bottom of the model domain) in the first column, the class name in the second and the class index in the third column. *Sleeping classes* are stratigraphy classes that are not part of the initial stratigraphy, but become added by a trigger of another stratigraphy class during the run. An example is a LAKE class that gets added above a GROUND class when enough surface water has accumulated. The SNOW class to be used by the run is selected in the fields snow_classname and snow_class_index. If no SNOW class is used, these fields can be left empty. NOTE: each stratigraphy class listed in STRAT_classes must be initialized separately in the parameter file (see stratigraphy classes).

S2.7 STRATIGRAPHY_STATVAR

STRAT_layers: used to initialize the initial depth profile of model state variable as layers with constant values. In the matrix, the variable names (that must match variable names in the stratigraphy classes) must be inserted in first line. The first row provides the start depth of each layer, and the last layer extends to the bottom of the model domain. NOTE: the state variables are in general provided in "human-readable" units. In the automatic initialization procedure, they are converted to the true unit of the state variable used in the simulations.

STRAT_linear: same as STRAT_layers, but values of the state variable for certain depths are provided. Between these depths, values are linearly interpolated.

S2.8 Stratigraphy classes – GROUND

These classes describe ground material consisting of mineral, organic, water, ice and air fractions. To enable the dynamic build-up of a seasonal snow cover by coupling to a SNOW class, each of the described classes has a twin called "CLASSNAME*_snow*" (e.g. *GROUND_freeW_seb_snow"*) which must be used instead. During the snow-free period, their behavior is identical to the class without the "…*_snow"* ending.

The compatibility of pairs of classes in the CryoGrid stratigraphy (ensured by interaction classes, Sect. 2.1.2) can be checked with the function "get_IA_class (*above_class, below_class*)" located in the folder (assuming he folder structure suggested in S1) CryoGrid_git/CryoGrid /source/TIER_2_full_classes/INTERACTION. Here, *above_class* is the name of the stratigraphy class located on above *below_class* in the CryoGrid stratigraphy. If this returns an interaction class (and not zero), the classes are compatible, e.g. *get_IA_class('SNOW_simple_bucketW_seb', 'GROUND_freeW_seb_snow')* returns the interaction class *IA_HEAT11_WATER10*, while *get_IA_class('SNOW_simple_bucketW_seb', 'GROUND_freeW_seb)* returns 0.

GROUND_TTOP_simple2: equilibrium TTOP approach (Sect. 2.2.2). Not compatible with lateral interaction classes (S2.12).
GROUND_freeW_ubtf: temperature boundary condition (Sect. 2.2.2), free water freezing characteristic (Sect. 2.2.3), no flow water balance (Sect. 2.2.4). Not compatible with lateral interaction classes (S2.12).

GROUND_freeW_seb: surface energy balance with scheme 2 for evapotranspiration (Sect. 2.2.2), free water freezing characteristic (Sect. 2.2.3), no flow water balance (Sect. 2.2.4). Compatible with lateral interaction class *LAT_HEAT* (S2.12).
GROUND_freeW_bucketW_seb: surface energy balance with scheme 3 for evapotranspiration (Sect. 2.2.2), free water freezing characteristic (Sect. 2.2.3), bucket scheme water balance (Sect. 2.2.4). Compatible with lateral interaction classes *LAT_HEAT*, *LAT_REMOVE_SURFACE_WATER*, *LAT_SEEPAGE_FACE*, *LAT_WATER_RESERVOIR* (S2.12).

GROUND_freezeC_seb: surface energy balance with scheme 2 for evapotranspiration (Sect. 2.2.2), Painter and Karra (2014) freezing characteristic (Sect. 2.2.3), no flow water balance (Sect. 2.2.4). Compatible with lateral interaction class *LAT_HEAT* (S2.12).
GROUND_freezeC_bucketW_seb: surface energy balance with scheme 3 for evapotranspiration (Sect. 2.2.2), Painter and Karra (2014) freezing characteristic (Sect. 2.2.3), bucket scheme water balance (Sect. 2.2.4). Compatible with lateral interaction classes *LAT_HEAT*, *LAT_REMOVE_SURFACE_WATER*, *LAT_SEEPAGE_FACE*, *LAT_WATER_RESERVOIR* (S2.12).

GROUND_freezeC_bucketW_seb_Xice: surface energy balance with scheme 3 for evapotranspiration (Sect. 2.2.2), Painter and Karra (2014) freezing characteristic (Sect. 2.2.3), excess ice representation (Sect. 2.2.5), bucket scheme water balance with

representation of standing surface water (Sects. 2.2.4, 2.2.5). Compatible with lateral interaction classes *LAT_HEAT*, *LAT_REMOVE_SURFACE_WATER*, *LAT_OVERLAND_FLOW*, *LAT_SEEPAGE_FACE*, *LAT_WATER_RESERVOIR* (S2.12).

GROUND_freezeC_RichardsEqW_seb: surface energy balance with scheme 4 for evapotranspiration (Sect. 2.2.2), Painter and Karra (2014) freezing characteristic (Sect. 2.2.3), Richards equation water balance (Sect. 2.2.4). Compatible with lateral interaction classes *LAT_HEAT*, *LAT_REMOVE_SURFACE_WATER*, *LAT_SEEPAGE_FACE*, *LAT_WATER_RESERVOIR* (S2.12).

S2.9 Stratigraphy classes - LAKE and GLACIER

These classes describe subsurface domains consisting of water and/or ice. To enable the dynamic build-up of a seasonal snow cover by coupling to a SNOW class, each of the described classes has a twin called "CLASSNAME*_snow*" (e.g. *GLACIER_freeW_seb_snow"*) which must be used instead. See S2.8 for compatibility of pairs of stratigraphy classes.

LAKE_simple_bucketW_seb: surface energy balance with scheme 1 for evapotranspiration (Sect. 2.2.2), free water freezing characteristic (Sect. 2.2.3), simple water body scheme with dynamic changes of water table (Sect. 2.2.7). Compatible with lateral interaction classes *LAT_HEAT*, *LAT_SEEPAGE_FACE*, *LAT_WATER_RESERVOIR* (S2.12).

GLACIER_freeW_seb: surface energy balance with scheme 1 for evapotranspiration (Sect. 2.2.2), free water freezing characteristic (Sect. 2.2.3), glacier mass balance scheme with surface meltwater automatically removed (Sect. 2.2.8). Compatible with lateral interaction class *LAT_REMOVE_SURFACE_WATER* (S2.12).

S2.10 Stratigraphy classes – SNOW

These stratigraphy classes are created by triggers in e.g. GROUND stratigraphy classes. For this to work, stratigraphy classes ending with "…_snow" must be used, e.g. *GROUND_freeW_seb_snow*. See S2.8 for compatibility of pairs of stratigraphy classes.

SNOW_simple_ubtf_mf: Constant snow density, temperature boundary condition and degree-day based melt model (scheme a, Sect. 2.2.6). Not compatible with lateral interaction classes (S2.12).

SNOW_simple_bucketW_seb: Constant snow density, surface energy balance and bucket scheme snow hydrology (scheme b, Sect. 2.2.6), meltwater automatically removed if it pools up above the snow surface. Compatible with lateral interaction classes *LAT_REMOVE_SURFACE_WATER*, *LAT_SEEPAGE_FACE*, *LAT_WATER_RESERVOIR* (S2.12).

SNOW_crocus_bucketW_seb: Snow microphysics, surface energy balance and bucket scheme snow hydrology (scheme c, Sect. 2.2.6), meltwater automatically removed if it pools up above the snow surface. Compatible with lateral interaction classes *LAT_REMOVE_SURFACE_WATER*, *LAT_SEEPAGE_FACE*, *LAT_WATER_RESERVOIR* (S2.12).

SNOW_crocus2_bucketW_seb: Snow microphysics, surface energy balance and bucket scheme snow hydrology (scheme c, Sect. 2.2.6), meltwater retained and allowed to pool up above the snow surface. This class is designed to be used with *GROUND_freezeC_bucketW_seb_Xice* which snowmelt water pooling above the surface transferred to the surface water pool after completion of snowmelt. Compatible with lateral interaction classes *LAT_REMOVE_SURFACE_WATER*, *LAT_OVERLAND_FLOW*, *LAT_SEEPAGE_FACE*, *LAT_WATER_RESERVOIR* (S2.12).

S2.11 LATERAL

LATERAL_1D: This lateral class is used to simulate interactions of a one-dimensional model domain with external environments/reservoirs (Sect. 2.3) in the TILE class "*TILE_1D_standard*". It must be used together with any of the lateral interaction classes described in S2.12. Note that *LATERAL_1D* must be set in *TILE_1D_standard*, even if there is no lateral interaction class selected.

S2.12 LATERAL INTERACTION

LAT_HEAT: lateral coupling to heat reservoir (Sect. 2.3.1).
LAT_REMOVE_SURFACE_WATER: surface water removal (Sect. 2.3.2).
LAT_OVERLAND_FLOW: overland flow (Sect. 2.3.2).
LAT_SEEPAGE_FACE_WATER: seepage face (Sect. 2.3.2).
LAT_WATER_RESERVOIR: water reservoir (Sect. 2.3.2).

S2.13 PROVIDER

The role of provider classes PROVIDER class is to organize the interactions between user input in e.g. parameter files and the RUN_INFO class. PROVIDER classes contain instructions to read the classes and their associated parameters from the parameter file and organize it in a standardized fashion, so that it becomes accessible for the RUN_INFO class. The PROVIDER class is selected by the variable *init_format* in *run_CG.m*. Three main initialization methods exist 1. *init_format* = 'EXCEL' and *init_format* = 'EXCEL3D' use spreadhseet-based parameter files, using the PROVIDER classes *PROVIDER_EXCEL* and *PROVIDER_EXCEL3D*, respectively (3D is used to initialize multi-tile 3D simulations with e.g. *RUN_3D_STANDARD*); 2. *init_format* = 'YAML' uses text files in yml-format as parameter files with the PROVIDER class *PROVIDER_YAML*; *init_format = 'MAT'* reads an already existing (typically created earlier through either 1 or 2 and stored in a file) PROVIDER classes from a mat-file with the class *PROVIDER_MAT*.

S3: Ancillary simulations for water redistribution during freezing

Here, we provide two additional benchmark simulations for the Mizoguchi (1990) experiment, using Richards equation to simulate the water balance. See Sect. 3.1.3.

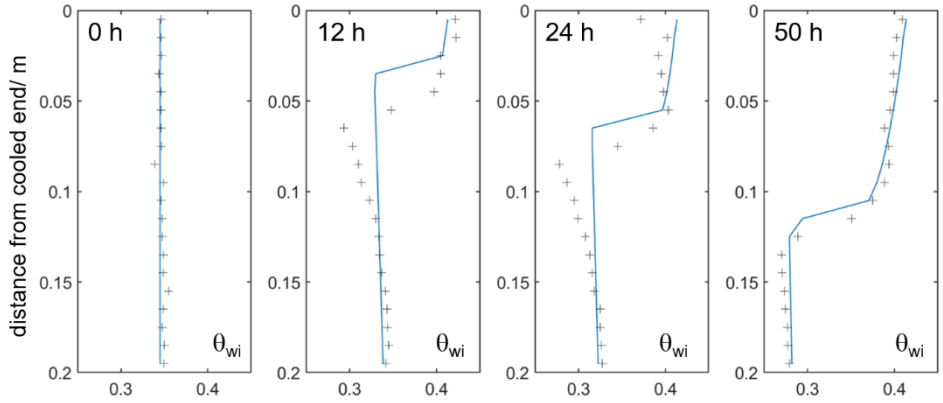


Fig S1: Simulated sum of volumetric water and ice content (blue lines) vs. measurements (crosses, digitized from Hansson et al., 2004) for the Mizoguchi (1990) experiment for experiment for 0, 12, 24 and 50 hours freezing time; linear heat transfer scenario, ice impedance factor calculated with $\Omega = 5$.

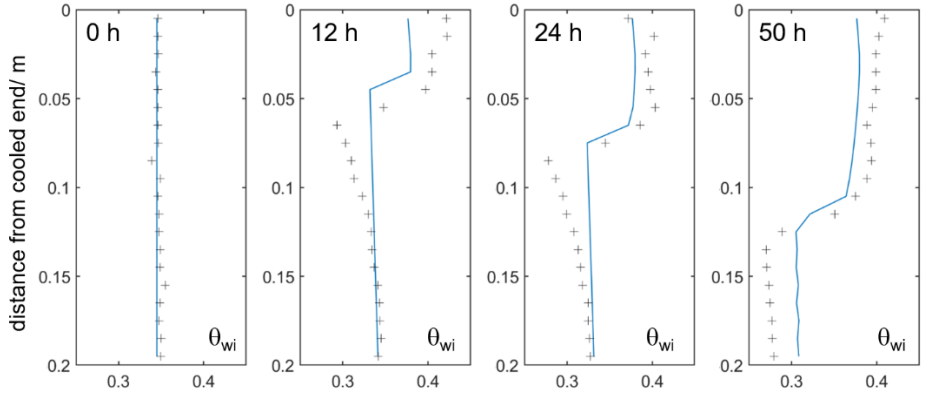


Fig S2: Simulated sum of volumetric water and ice content (blue lines) vs. measurements (crosses, digitized from Hansson et al., 2004) for the Mizoguchi (1990) experiment for 0, 12, 24 and 50 hours freezing time; nonlinear heat transfer scenario, ice impedance factor calculated with $\Omega = 7$.

S4: Parameter files and model forcing

For all the simulations in Sect. 3.2, we provide parameter files in spreadsheet format (as well as the model forcing) in the file "CryoGridCommunity_parameter_files.zip" on https://doi.org/10.5281/zenodo.6522424. The parameter files are organized in the folder structure suggested in S1, so that the simulations can be directly started when following the steps outlined in S1. After unpacking "CryoGridCommunity_parameter_files.zip", all subfolders (i.e. "forcing", "glacier", etc.) must be copied to the folder "CryoGridCommunity_results" (see S1). To start a particular simulation, the variable "run_name" must be set in "run_CG.m" located in the folder in "CryoGridCommunity_run". In the following, the run names employed to create the simulations evaluated for the figures in Sect. 3.2 are provided in italics:

Figure 10. Black line: *reference_run_snowfall_100_percent*; broken blue line: *reference_run_snowfall_90_percent*; solid blue line: *reference_run*

Figures 11-13. *reference_run*

Figure 14. Blue: *reference_run.xlsx*; green: *reference_run_sand;* red: *reference_run_free_water_freezing*

Figure 15. Blue, solid line: *reference_run*; blue, dashed line: *reference_run_overland_flow*; green: *reference_run_no_drainage*; red: *reference_run_inflow_of_water*

Figure 16. *reference_run_overland_flow*

Figure 17. Blue: *reference_run*; green: *reference_run_water_balance_constant*; red: *reference_run_water_balance_Richards_equation*

Figure 18. Left: *reference_run*; middle: *reference_run_water_balance_constant*; right: *reference_run_water_balance_Richards_equation*

Figure 19. *reference_run_water_balance_Richards_equation*

Figure 20. Blue: *reference_run;* green solid: *reference_run_constant_density_snow_250_kgm-3*; green dashed: *reference_run_constant_density_snow_275_kgm-3*; red solid: *reference_run_crocus_snow_normal*; red dashed: *reference_run_crocus_snow_arctic*

Figure 21. Blue: *reference_run*; red: *reference_run_temperature_boundary_condition*; green dashed: *TTOP_entire_period*; green solid: *TTOP_two_year_periods*; green dotted: *TTOP_as_upper_boundary*

Figure 22. Blue: *reference_run;* red: *water_body*; green: *glacier*

Figure 23. *water_body*

Figure 24. *glacier*

References

Hansson, K., Sımunek, J., Mizoguchi, M., Lundin, L.-C., and Van Genuchten, M. T.: Water flow and heat transport in frozen soil: Numerical solution and freeze–thaw applications, Vadose Zone Journal, 3, 693–704, https://doi.org/10.2113/3.2.693, 2004.

Mizoguchi, M.: Water, heat and salt transport in freezing soil, Ph.D. thesis, University of Tokyo, 1990.

Painter, S. L. and Karra, S.: Constitutive model for unfrozen water content in subfreezing unsaturated soils, Vadose Zone Journal, 13, https://doi.org/10.2136/vzj2013.04.0071, 2014.