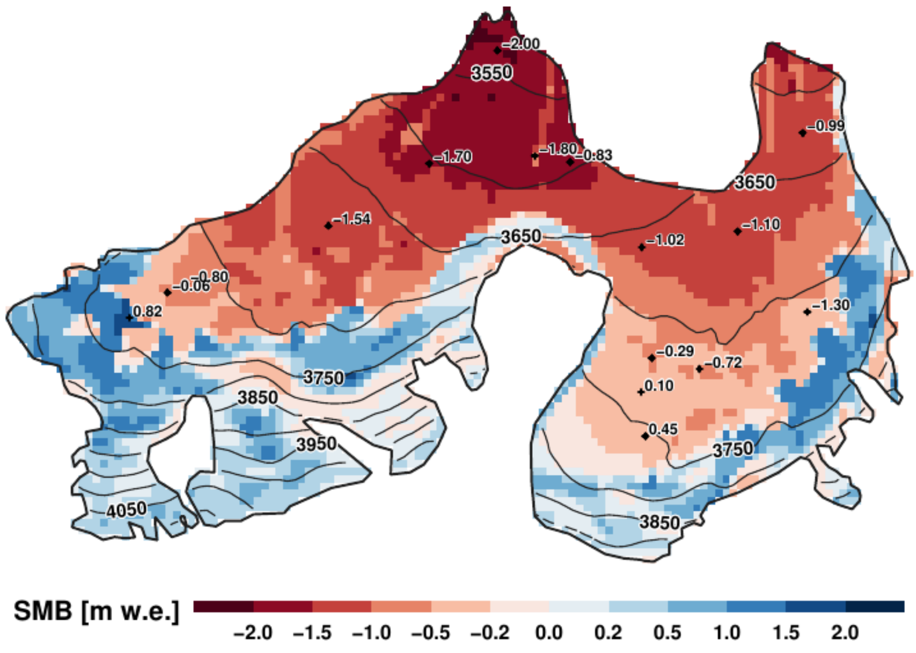
UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

Simulation of glacier mass balance from point measurements



Enrico Mattea

enrico.mattea@unifr.ch

June 2021

Chapter 1

Overview

The purpose of this software package is to model and visualize the distributed annual mass balance of a glacier. The simulation is driven by a daily meteorological series of temperature and precipitation, and is constrained by point measurements of melt (and optionally of winter snow accumulation).

Chapter 2

Model description

The simulation is performed on a **gridded domain**, defined by a digital height model (DHM). The mass balance model is driven by a **daily meteorological series** of temperature and precipitation: starting from an initial condition of snow-water equivalent (SWE), the model computes daily accumulation and melt for each grid cell, keeping track of the accumulated mass balance. **Each year is simulated individually**: the simulated period for a year is defined to include the whole observation period of the stakes measured on that year, as well as the whole hydrological year (1 October-30 September) and also possibly a user-defined period. So in general the simulated period for a single year will be more than 365 days, and in a multi-year simulation there will be some overlap (this is not an issue since the final mass balance is extracted at the correct dates). The simulation of different years can use different input height models and glacier outlines: this is useful if the glacier shape or topography change during the modeled years. There is also an option to automatically **interpolate between elevation grids** of different years.

The **initial snow condition** for a year is computed from topography (elevation, slope, curvature), from a user-defined snow line altitude and elevation gradient, from an avalanche simulation and (if available) from the winter snow probes. From the second modeled year there is the option to use the previous model output as starting condition.

Distributed **daily melt** is computed using an enhanced temperature index (degree-day) model (Hock, 1999), considering both mean daily air temperature and potential solar radiation. Different radiation factors are used for snow, bare ice and debris-covered ice. Additionally, there is the option to use a spatially variable melt factor for ice, to account for darker ice towards the glacier terminus. Distributed **daily accumulation** is derived from precipitation, using a rain/snow temperature threshold and a distribution grid similar to the one for the initial snow condition.

The model features a **process-based avalanche routine** (Gruber, 2007) to redistribute the snow mass. Avalanches are simulated at user-defined fixed dates; they can change the net amount of snow on the glacier surface if an avalanche enters/exits the glacier surface (this is why the entire grid is simulated and not only the glacier cells). Elevation grids are **automatically pre-processed** to ensure a **correct hydrologic flow** of the avalanche: all grid sinks are filled using 4-connectivity (that is, from each grid cell there is a strictly descending path to the grid border going through the four surrounding neighbors). Still, for more control it can be a good idea to use elevation grids where the sinks are already filled.

The model for each year is automatically run several times to **optimize the parameters**, in order to find the best fit to the measurements. This optimization works **by comparing the stake measurements to the modeled**

mass balance at the location of each stake (bilinearly filtered from the grid). The comparison is performed over the exact observation period of each stake (so multiple observations during a year can be used). The parameters are tuned to **cancel the mean stake bias** (the RMS error is not considered). If winter measurements are available (snow probes), the model is first run over the winter period only, to optimize a factor of **precipitation correction** against the winter stake measurements. Then the model is run over the entire period (longer than a year, as described above) to optimize the **melt factors** against the annual stake measurements. The model also supports **snow depth measurements with an unknown starting date**: the starting date is automatically set to the date of the previous annual mass balance minimum. This can be used to include summer measurements from snow pits and snow probes. If the measured stakes are not uniformly distributed over the glacier, the optimization can be biased towards a certain glacier area. To mitigate this, the model can **cluster together stakes** which are closer than a distance threshold, treating them as a single stake.

After optimization over the global bias, the model performs a **final local correction of the simulated mass balance** within user-defined **elevation bands**: for each elevation band it computes the local model bias compared to the stakes in that band, and corrects the model output accordingly (with a linear interpolation between bands, for a smooth result). Still, the bands do **not** need to cover all the vertical extent of the glacier or all the stakes: for example, you can have band boundaries at *(3000, 3300, 3600, 3900) m* and a stake at 4000 m. Then the stake is not included in any band and it will not contribute to the bias correction. Above the highest band and below the lowest band the local bias always decreases linearly to zero. The local correction accounts for differences between the model result and the contour-line method. The local correction is initially computed as a single map, with a correction relative to the whole measurement period. Later, **it is also converted into a daily time series of mean corrections**. This is done to compute a time series of mean glacier-wide mass balance including the correction. To compute the correction time series, the mean correction at the end of the measurement period is distributed uniformly over the melt amounts, so that the corrected mass balance deviates progressively from the original one as cumulative melt increases.

Chapter 3

Installation

The model is written in the free, multi-platform programming language R. It was developed under R version 3.6.3 and has not been tested with other versions. To run the model you need to install (just once):

- the **R language interpreter** (download page)
- a **code compiler** to install some additional R packages (see here)
- the **R packages** used by the project. These are listed in Table 3.1; most of them also depend on additional packages which are automatically installed as required. Package installation requires an internet connection and can take quite some time. All packages can be installed from within R using these two commands:

  ```
  > install.packages(c("ggpubr", "gstat", "metR", "remotes", "Rfast", "RStoolbox", "spatialEco",
  "topmodel", "timeSeries"))
  > remotes::install_github("coolbutuseless/ggpattern")
  ```

  More details on the installation of R packages can be found here.

It is strongly recommended to use the RStudio IDE (download page) to run the model, although you can also run it from the R command line or any other IDE.

For performance reasons one routine (the process-based avalanche simulation, in file *func_avalanche_gruber.cpp*) is written in C++ and is automatically code-compiled when you run the model for the first time. A pure R implementation of the same avalanche function is also provided (*func_avalanche_gruber.R*), but it is not recommended because it is much slower, especially if the domain includes large areas of steep avalanche terrain. If you still want to use the pure R version you can simply set *avalanche_routine_cpp* to FALSE in the *set_params.R* file.

Package name	**Source**	**Purpose**
cowplot	CRAN	Align plots
ggplot2	CRAN	Base plotting library
ggpubr	CRAN	Multi-page PDF
gstat	CRAN	Spatial interpolation of snow probes
metR	CRAN	*geom_text_contour()*
raster	CRAN	Grid manipulation
Rcpp	CRAN	Avalanche function implemented in C++
remotes	CRAN	*install_github()*
reshape2	CRAN	*melt()*
Rfast	CRAN	*rowSort()*
RStoolbox	CRAN	*ggRGB()*
scales	CRAN	*rescale()*
stringr	CRAN	*str_split()*
sf	CRAN	*st_read()*
sp	CRAN	*SpatialPolygons()*, for glacier outlines
spatialEco	CRAN	*curvature()*
topmodel	CRAN	*sinkfill()*
timeSeries	CRAN	*interpNA()*
ggpattern	GitHub	Pattern as histogram fill

Table 3.1: R packages explicitly loaded by the project

Chapter 4

Workflow

4.1 Input files

This section describes the model input preparation. Required input data consist of:

- one or more elevation grids
- one or more grids of surface type (ice, firn, debris)
- 365 grids of daily solar radiation
- one or more glacier outlines in vector format
- a text file with the daily meteorological series
- one or two files with the annual (and optionally winter) point measurements of mass balance

Supported **grid file formats** can be listed using R command `rgdal::gdalDrivers()`. Common choices include GeoTIFF and ESRI/ASCII grid. **All grids** (elevation, radiation and surface type) should have the **same coordinates system, extent and resolution**. Grids of a same variable should also use the same file format. The **grid extent** should be wide enough to include a margin of some row/columns around the glacier. Ideally, the whole water catchment of the glacier should be included (for avalanches which can reach the glacier from outside). **Grids should not have any missing (NA/NaN/nodata) value.**

If available, **multiple grids** (max one per year) can be provided in order to run the simulation on up-to-date input data: for example repeated digital elevation models. The grid years do not have to match: one could supply several elevation grids but just a single file for the surface type. File names should follow a constant pattern: *<prefix><year><suffix>*, where *<prefix>* and *<suffix>* can be freely chosen but have to be the same for a same variable. For example, with two elevation grids and one surface type grid, file names could be *dhm_2015.tif* and *dhm_2018.tif* for elevation, and *surface_type2017.tif* for surface type. In this case prefixes would be *dhm_* and *surface_type*, suffixes would be *.tif*.

For **elevation** there is the option to linearly interpolate the grids for the missing years (else the grid which is closest in time to the modeled year will be selected). The model will process the elevation grids to obtain a hydrologically correct map, with no sinks. For more control you can provide elevation grids where the sinks have already been filled. Be aware that the linear interpolation between two sink-filled grids could still produce sinks, which will be removed by the model.

Grids of **surface type** use the codes reported in Table 4.1. Note that the glacier cells are derived from the vector outline (see below): therefore there shall be no grid cell with code 4 (non-glacier area) inside the glacier outline, because the simulated mass balance of this cell would be wrong (melt would stop as soon as there is no more snow on the cell). **To produce an accurate grid of surface type**, it is best to start from a grid of non-glacier (value 4 everywhere), then use the glacier outline as a mask to put the value 0 over the glacier, and then draw other vector masks to change the value of firn and debris cover. Under QGIS, the relevant tools are

```
Layer → Create Layer → New Temporary Scratch Layer
Raster → Conversion → Rasterize
Raster → Raster Calculator.
```

Code	Meaning
0	Bare ice
1	Firn
4	Non-glacier areas
5	Debris-covered ice

Table 4.1: surface type codes

The grids of **daily radiation** are used within the melt model. Radiation should be expressed in W m^{-2}. You should provide just 365 daily grids (inter-annual variability of radiation is ignored). On leap years, the model replicates the 365th day twice. The grids can be generated using any tool. Under the *utils/* directory there is an R script to generate them using the SAGA GIS command line interface (explained here; installation instructions here). Some minor adaptations may be needed to run the script on Windows and Mac OS X systems.

Glacier outlines in vector format are used to select the grid cells which contribute to mass balance, and also to plot the output. Like the grids of elevation and surface type, you can supply several outlines which correspond to different years; in this case, the same naming rules apply (file names could be for example *outline_2015.shp* and *outline_2018.shp*). Outlines can use either the ESRI shapefile format (.shp) or the XYZN format.

The **meteorological series** for the whole simulation should be provided as a single text file with 5 columns (space- or tab-separated): year, day of year (1-366), hour (ignored), temperature (in °C) and precipitation (in mm w.e.). The model will select the appropriate period for each modeled year. The series should be long enough to cover the full simulation period, which is usually longer than the period of stake observations as it includes the hydrological year.

Mass balance observations should be provided as one text file for annual measurements, and optionally another for winter measurements. The files should have 8 columns (space- or tab-separated): point name, start of observation period, end of observation period, X coordinate, Y coordinate, Z coordinate (altitude), stake measurement in cm and density in g cm^{-3}. The observation periods should be specified as DD.MM.YYYY, and the coordinates should use the same system as the grids. One **special value for the start of the measurement period** is **NA**: it is interpreted by the model as **"end of the previous melting season"** and should be used for snow pits and snow probes, which measure down to the previous summer surface (see Chapter 2). If the stake measurement is known already as water-equivalent (and not stake height + density) you can simply set the density to 1. Point measurements on the glacier edges can be problematic due to bilinear filtering (non-glacier cells could contribute to the model estimation of mass balance for a stake). To prevent this issue you should always have at least one full grid cell between the measured point and the glacier edge.

If you are running the model several times (for example looking for the best parameters), you can **speed up the data loading** (which is quite slow for the radiation files) by using a **boot file**: in the code file *main.R* you can set *boot_file_write* to TRUE, then the model will generate a *.RData* boot file. For the next model runs you can set *boot_file_read* to TRUE (and *boot_file_write* back to FALSE) and the model will load the boot file (very fast) instead of all the single data files. If you make a change to the input data (see list at the beginning of this section), then you will have to re-load the data again (with *boot_file_read* set to FALSE) and generate again the boot file. Instead if you change a model parameter (see Sect. 4.2) you don't have to re-generate the input file. You can use the boot file also to share a certain model input with other model users (in this case you probably want to share also the parameter files: see next section).

4.2 Model parameters

The model has **two groups of parameters**: one group which is fixed for the whole model run (over all years), and one group which changes each year (annual parameters).

The **fixed parameters** are set by editing file ***set_params.R***. There, each parameter is explained in the file comments. The file is actually an R code file: each value should be followed by a comma (","), multiple values are specified as *c(<value1>,<value2>,<value3>)*. You can keep *c(<value1>)* if a parameter can have multiple values but you use only one. For example, if you have elevation grids for 2012 and 2015 you could set *dhm_years = c(2012,2015)*; then if you choose to use only 2012 you can leave *dhm_years = c(2012)*. Finally, TRUE/FALSE values should be in UPPERCASE. See here for a quick reference on R syntax.

Apart from the file paths, there are some **glacier-specific parameters** in *set_params.R*, which you should always change from one glacier to another:

- the glacier name (*name_glacier*)
- the reference elevation of the weather series (*weather_aws_elevation*)
- the cut-off elevations for snow distribution (*weather_max_precip_ele* and *elevation_effect_threshold*)
- the initial snow-line elevation (*initial_snowline_elevation*)
- the model years (*first_year* and *last_year*)

Other parameters in *set_params.R* affect the physical simulation. You can tune them (running the model several times and manually checking the result) until you find a good agreement with the measured stakes (low RMS error). The effect of each parameter is usually quite small; still, changing the avalanche parameters can have a very large impact if it causes an avalanche deposit to reach/miss a stake. This also implies that **the avalanche routine can interfere with the model optimization:** the reason is that the model bias can depend on whether a measured stake is reached or not by an avalanche, and the avalanche deposit length depends on the involved mass, which in turn depends on the model parameters which are being optimized). To solve this, there is an option (*optim_max_iter*) which stops the optimization after a given number of iterations (by default 20), even if convergence was not reached. If this happens, you should check that the model output makes sense, and possibly experiment with different model parameters (such as *deposition_mass_lim*).

The **annual parameters** should be provided as text files (one file per year, with a consistent naming scheme: *<prefix><year><suffix>*). They include the precipitation correction and melt factors: these are optimized during the model run (see Chapter 2), so the values in the parameter files are only the starting values. They are useful to set the **relative magnitude** of the melt parameters (which are optimized together, so their ratio remains constant). Precipitation correction is a multiplication factor, so 100 % means no correction. Parameters *evaluate_snowdist* and *year_snowdist* are kept for compatibility purposes and are ignored by the model. Parameter *mb_corr_ele_bands* defines the boundaries of the bands for the **final local correction of mass balance** (see last paragraph of Chapter 2). You should carefully select these band boundaries to **include at least one stake in each band** (else the correction does not make sense).

4.3 Model run

After setting up all input and annual parameters, the model can be run by simply executing the *main.R* file (from RStudio: select all lines and click on `Run`, or use `Ctrl+A` and `Ctrl+Enter`). You can use a boot file to speed-up data loading (last paragraph of Sect. 4.1).

4.4 Output files

Model output is written to a directory called *output/<name_glacier>/*, where you can define *<name_glacier>* in file *set_params.R*. For each simulated year, the following output is computed:

- Mass balance map over the hydrological year (from 1 October of the previous calendar year to 30 September of the current calendar year).
- Mass balance map over the "annual measurement period". This is defined as the period between the first annual stake observation, at the beginning of the melt year (usually in August-October of the previous calendar year), and the last stake observation at the end of the melt year (usually in August-October of the current calendar year).
- Mass balance map over the annual measurement period, including the local correction with elevation bands. This is called **"Measurement period (annual, corrected)"** or equivalently "Annual, final".
- Mass balance map over a user-defined annual period (called "fixed" period).
- Mass balance map over a user-defined winter period.
- (Only if winter measurements were supplied): mass balance map over the "winter measurement period". This is defined as the period between the first winter stake observation (usually in October of the previous calendar

year) and the last winter stake observation (usually in April of the current calendar year).

- Daily **time series of the cumulative mean mass balance** over the glacier. The series includes the original mass balance (as simulated by the accumulation/melt model) as well as the mass balance corrected with the elevation bands (see end of Chapter 2), and also the separate components of melt and accumulation.
- Daily time series of the cumulative mass balance simulated at the stake locations.
- Annual and winter **mass balance** values classified **into elevation bands**.

Of the output above, the maps are written to individual grid files, the other values to plain-text comma-separated-value (CSV) files. Moreover, **all the output is plotted** to files in vector PDF format.

The model also summarizes the results of the whole run into **overview files**. If requested, the model can also generate daily plots of snow-water equivalent and cumulative mass balance. This can be useful to investigate unexpected mass balance distributions and patterns, but it makes the processing very slow.

Chapter 5

Code architecture

The model code is organized into function files, stored under *functions/*. Moreover, under *procedures/* there are several code blocks which are called directly by *main.R*. You don't need to modify these functions and procedures while using the model; still, if you are interested, in these source files you can find very frequent comments explaining in detail the inner workings of the model. Within the main loop, the model output after simulating a year is stored in a large list called *mod_output_annual_cur*.

Bibliography

Gruber, S.: A mass-conserving fast algorithm to parameterize gravitational transport and deposition using digital elevation models, Water Resources Research, 43, https://doi.org/10.1029/2006WR004868, URL http://doi.wiley.com/10.1029/2006WR004868, 2007.

Hock, R.: A distributed temperature-index ice- and snowmelt model including potential direct solar radiation, Journal of Glaciology, 45, 101–111, https://doi.org/10.3189/S0022143000003087, URL https://www.cambridge.org/core/product/identifier/S0022143000003087/type/journal_article, 1999.