README: Simulating Temperature Transients in First Year Sea Ice

Nitay Ben-Shachar

November 16, 2020

This package solves a 1D transient thermodynamic model for the temperature profile of sea ice/snow which can be solved to arbitrary resolution both spatially and temporally, coded in MATLAB. The growth of sea ice is determined by balancing heat fluxes at the ice-ocean interface. An oceanic heat flux and snow layer can be incorporated as specified by input files but snow is assumed to have constant density. A sharp interface between the ice and ocean is assumed. Salt transport is accounted for only in its variation of the ice's physical parameters.

This program was coded and tested with MATLAB R2019b and possibly uses functionality not available from earlier releases.

1 Model Details

Existing sea ice transient temperature models in the literature, such as (Maykut and Untersteiner, 1971), (Launiainen and Cheng, 1998), (Bitz and Lipscomb, 1999), use a one-dimensional model to simulate temperature transients. The model simultaneously solves for the growth rate of the sea ice and the temperature profile in the sea ice and a possible snow cover. This is done by balancing heat throughout the sea ice, snow and at each interface, subject to the user specified forcing factors. The numerical solution is attained by freezing the moving boundary of the sea ice and snow using a Landau transform, and using the Method of Lines to discretize over depth. Both techniques are briefly detailed below, but are described in more details in the manuscript.

Freezing the Boundary is a mathematical method especially useful in analysing systems with moving boundaries. It has been incorporated in this model for mathematical ease and elegance. Rather than adding simulation calculation points as the ice grows, we solve an equivalent mathematical problem over a fixed depth interval by defining the depth-like variables $\chi, \xi = \frac{z}{H(t)}$ where $H(t)$ is the ice/snow depth. By definition, the slab of sea ice/snow always lies in corresponding values of χ, ξ between 0 and 1, and so solving the (modified) heat equation in terms of χ, ξ takes into account any sea ice growth or snow dump/melt.

In using the Method of Lines we first discritizing over the depth variables ξ, χ. The partial differential equation is then converted to a set of coupled ordinary differential equations, each describing the temperature evolution at a fixed value of χ, ξ. The temperature profile is linearly interpolated over time and space from the solutions of each ordinary differential

equation. The spatial resolution of the model can be chosen to be arbitrarily high/low, with discretizing at ≈5 cm intervals usually being sufficiently accurate.

See the manuscript for details of the model and references for the formulae.

2 Using the Package

In order to run the program, at least one input file must be provided in the input folder. All input forcing factors (to be placed in the `/input` folder) files are linearly interpolated between the given points in time/depth, with depth/time in the left column and the corresponding value in the right column. Time is provided in days and depth in meters. The essential input file is

airTemps.csv: Air temperatures (°C) over time.

Further forcing factors may be specified using the following files.

salinity.csv: Salinity (in grams per gram) profile over depth and assumed static in time. The depths should all be negative numbers, corresponding to depths below the sea level. Default value: 5 ppt = 5×10^{-3} g/g.

snow.csv: Snow depth over time. Default value: 0 m.

oceanFlux.csv: Ocean flux over time. Default value: 0 W/m^2.

longWaveRadiation.csv: Longwave radiation loss of the upper surface to the atmosphere over time. Default value: Net longwave radiation is set to 0 W/m^2.

location.csv: Longitude and latitude over time. Default value: No short wave radiation included.

windSpeed.csv: Magnitude of windspeed over time. Default value: 5 m/s.

humidity.csv: Specific humidity of the atmosphere. Default value: 10^{-4}.

bulk_richardson.csv: Bulk Richardson number of the atmosphere. Default value: 0.

airPressure.csv: Air pressure over time. Default value: 1 atm = 101325 Pa.

cloudiness.csv: Cloudiness factor over time. Default value: 0.

As well as the above files, some parameters should be adjusted in the file seaIceModel.m. These values are clearly labeled near the top of the file.

num_of_points_snow/ice: These constants determine the number of calculation points that are used in discritizing over the depth of the snow/ice for the simulation. A constant number of points is used meaning that the resolution of the model changes over time, for example, setting num_of_points_ice = 10 for a simulation were the ice slab grows from -0.2 m to -0.5 m means that the model's resolution

changes from 50 points/meter to 20 points/meter. If the output temperature profiles seem unrealistic, for example with sharp changes, a likely solution would be to increase the spatial resolution of the simulation by increasing these constants.

opts: This determines the accuracy that the ODE solver takes, so determines the temporal accuracy of the simulation. MATLAB's stiff ODE solver `ode15s` is used to determine the numerical solutions to the temperature transients of the sea ice/snow and choose a time resolution that corresponds to an arbitrarily accurate solution specified by odeset. 'RelTol' determines the percentage error tolerated for each variable. 'AbsTol' determines the absolute numerical error allowed for each variable. See the MATLAB documentation for further settings that could be selected.

final_t: This is the number of seconds that the solver should run for. Select the duration for which the solution is desired. Time being zero is set to the first line in each input file (airTemps.csv, snow.csv).

time_offset: The solver will begin the solution from the point time_offset, with t=0 being defined at the beginning of the input files.

initial_depth: This is the initial depth of the ice, at the point in time specified by time_offset. This value should be negative and measured in meters.

present_dt: This is the temporal resolution (in seconds) in which the solutions will be reported.

z_res_ice: This is the spatial resolution (in meters) in which the sea ice temperature profile solutions will be reported.

z_res_snow: This is the spatial resolution (in meters) in which the snow temperature profile solutions will be reported.

Note: Increasing present_dt, z_res_ice and z_res_snow does not increase the accuracy of the solutions, only the resolution in which the solution is reported. To increase the accuracy increase the number of calculation points in the model and decrease the error tolerance (odeset).

start_date: This is the simulation start date (and time) set as a `datetime` structure, required for the calculation of the solar radiation.

T_freezing: This is the freezing temperature of the sea water, taken to be constant. This can be adjusted if the salinity of the water is well known. Further, it can be extended into a subroutine that adjusts the freezing point depending on the salinity, depth or any other variables of interest.

V_a: This is the proportion of incorporated air in the freezing sea ice, taken to be constant. Again, this can be made to depend on any number of parameters via a subroutine.

Once all the necessary variables have been provided, with the MATLAB working directory set to the folder containing the file seaIceModel.m, run the simulation by typing model_output = seaIceModel() into the MATLAB command line. The solutions will be stored in the variable model_output as detailed here:

model_output.time: A 1D array containing the values of time (in seconds), starting from 0 corresponding to the time of time_offset, at which the remainder of the solutions are given. This temporal resolution can be adjusted with the constant present_dt.

model_output.z_ice: A 1D array containing the depths associated with the values presented in the temperature profile of ice.

model_output.z_snow: A 1D array containing the depths associated with the values presented in the temperature profile of snow.

model_output.temp_profile_snow: A 2D array containing the solution to the temperature profile in the snow as a function of time. The temperature profile at a specific time is given by temp_profile_snow(time_index, :) where time_index is an integer corresponding to the time of interest. The temperature profile is evaluated at time increments of present_dt and so time_index=$\frac{\texttt{time}}{\texttt{present\_dt}}$. Points in the air are filled with a dummy value of NaN.

model_output.snow_depth: A 1D array containing the snow depth over the desired times. This will merely be an interpolation of the provided snow thickness at the points of interest.

model_output.temp_profile_ice: A 2D array with the corresponding solution of the ice temperature profile at the times of interest, similar to temp_profile_snow. Points in the water are filled with a dummy value of the freezing temperature.

model_output.ice_depth: A 1D array containing the solution to the ice depth over time.

The time listed in the first column of each input file is matched with the simulation time of $t = 0$. `time_offset` may be used to shift the start of the simulation time along the input timeline.

Snow is completely controlled by the input file and as such the solver will raise a warning if temperatures above the melting temperatures are reached. Melt occurring not at the bottom surface of the sea ice is not considered in this model. Further, setting the snow cover to 0 m causes the solver to fail, and as such a small value (default threshold is $10^{-4}\,^{\circ}$C) is adopted instead. Snow covers below this threshold are not considered in the computation of the surface energy balance and solar radiation.

3 Troubleshooting

3.1 Unable to reduce time step below minimum amount

If a MATLAB warning that the minimum time step was hit while trying to solve the ODEs, the resolution of the simulation must be increased by increasing the number of calculation points num_of_points_ice/snow.

3.2 Sharp jump in the snow temperature profile

This is due to the solver completely jumping over a critical time where the snow depth changed rapidly. This can be fixed by reducing the error tolerance or adding a minimum time step being a fraction of the length of the rapid snow melt/dump, however, these methods can make the solver run for a long time. If only one major jump is present, the solver can be separated into three solutions, where the first runs from the beginning to just before the major jump and uses a large minimum time step. The second covers the small region associated with the large jump using a small maximum time step, and the last covers the remainder of the simulation with a large maximum time step. Each solution would use the final output of the previous as its initial variables, and all three solutions are stitched together. See the MATLAB documentation for odeset for possible ways of increasing the solver accuracy and ensuring that it doesn't skip the points of interest.

3.3 Failure of Solver - Snow Cover

Sometimes the solver fails at a point when the snow depth changes from nonzero to zero and vice versa. A resolution to this is to simulate the regions of non-zero snow depth separately to regions of zero snow depth. Another resolution is to simulate a very thin snow depth in times of negligible snow depth.