IceBern2D: An Efficient Two-Dimensional Ice Sheet Model for Paleoclimate Studies.

Master's thesis

Faculty of Science

University of Bern

presented by

Basil Neff

2014

Supervisor:

Prof. Dr. Thomas Stocker
Physics Institute, Climate and Environmental Physics
and Oeschger Centre for Climate Change Research

Advisor:

Dr. Andreas Born
Physics Institute, Climate and Environmental Physics
and Oeschger Centre for Climate Change Research

Abstract

To understand the climate system of the past 100,000 years and longer it is important to investigate the role of the ice sheets. They actively interact with the rest of the climate system and have direct and indirect influences.

In this study, we developed the vertically integrated ice sheet model IceBern2D. This model was built to understand the two dimensional flow of an ice sheet at a high temporal and spatial resolution under different climatic conditions. The ice flow in the IceBern2D model is integrated over the entire column and together with all other values uniform at each grid point. The climate forcing of the model requires temperature and precipitation at a daily resolution only. Ablation is calculated by an empirical positive degree day (PDD) factor β. The precipitation below a specific temperature threshold is considered as accumulation. The bedrock is isostatically adjusted to the pressure of the overlying ice. Sea level is adapted to the captured ice masses on land.

All empirical parameters are varied to determine the sensitivity of the ice distribution and volume to these variations. The parameter combinations build three groups with different climate forcing and 180 members each.

We show that the model produces realistic results with simplified climate forcing. The ice distribution in the northern hemisphere with a constant last glacial maximum (LGM) climate forcing is comparable with LGM reconstructions. However, the ice volume is overestimated and is not consistent with LGM conditions.

The sensitivity study shows that the PDD factor β which quantifies the melting of ice is the most important for total ice volume. This empirical factor has a high dependency on the different density (ice vs. snow) and location. The model does have one state of the ice only and β does not change with its location.

We can show that the adjusting bedrock leads to oscillations of the Laurentide ice sheet with a constant climate forcing. Other ice sheets reach a constant equilibrium after their build-up. This oscillation is caused by an interaction between the bedrock relaxation time, surface mass balance and the topography. The basic interaction between these processes is known but not all details are yet fully understood. The sea level adjustment has an important influence on the ice distribution.

The model is efficient and well-suited for simulations of ice distributions on large temporal and spatial scale.

Contents

1 Introduction

The Role of Ice Sheets in the Climate System

To understand the climate system of the past 100,000 years and longer it is important to investigate the role of the ice sheets. They actively interact with the rest of the climate system and have direct and indirect influences. Changes may cause additional feedbacks and occur in ocean surface temperature, ocean circulation, continental water balance, atmospheric ciruclation, the extent and types of vegetation and land-surface albedo (Clark et al., 1999).

Large ice sheets as the Laurentide ice sheet play an important role in their interaction with the atmosphere. Atmospheric jets are stronger in the Atlantic sector at a time of maximal land ice extent during the last ice age compared to present-day climate (Li and Battisti, 2008). The topography of the ice sheet controls the overall placement of high and low pressure centers (Pausata et al., 2011; Merz et al., 2013a). Substantial change of the albedo at areas where the land turns to ice leads to a shift in the Earth's energy balance (Cess et al., 1991).

The growth of ice also leads to the reorganization of watersheds and a re-routing of continental runoff. In addition to this spatial redistribution, water drainage into the ocean becomes discontinuous as periods of relative quiescence are typically followed by rapid ice-wasting events with potentially high impact on the ocean circulation. The probably best-known and largest of such local deglaciations are Heinrich events (Heinrich, 1988) that may reduce the thermohaline circulation drastically and produce an intense warming in both hemispheres (Crucifix and Berger, 2002). Freshwater experiments in the North Atlantic with a comprehensive climate model show a strong reduction of the Atlantic meridional overturning circulation and a warming in the South Atlantic (Stocker et al., 2007). The meridional overturning was nearly, or completely, shut down during the deglaciation in the North Atlantic region beginning with the iceberg discharge Heinrich event H1, 17,000 years ago (McManus et al., 2004).

The Laurentide ice sheet in the center of the North American continent had the greatest influence on the sea level change at the last glacial maximum (LGM). Around 80 m of the sea level change are contributed from the North American ice sheet and 18 m from the Eurasian ice sheet. Overall the northern hemisphere is the dominant component sea level variations and contributes more than 100 m to the 130 meters of LGM sea level lowering

(Clark and Mix, 2002). The influence of the ice sheet during the LGM on the climate is confined almost exclusively to the Northern Hemisphere (Broccoli and Manabe, 1987). Therefore, the focus of the IceBern2D model is on the northern hemisphere.

What Drives the Changes in the Ice Sheets?

There are two plausible theories about the source of the glacial cycles of the last 3 million years. The classical theory posits that the change of the insolation, in particular its seasonal variations, is the driver of the ice age cycle (Milankovitch, 1941). The second theory emphasizes the role of variations in atmospheric CO_2 levels and associated radiative forcing, as suggested by reconstructions from ice cores (Siegenthaler et al., 2005). With the current knowledge it is impossible to distinguish which process is more important for which time frame (Paillard, 2006). Most likely, the combination of the two processes, including their independence, is required to quantitatively bring about ice age cycles.

Challenges for Climate Models

The climate system is complex and difficult to understand in its entirety. Even if we knew all involved natural and anthropogenic processes and could reproduce them in a model, there would be at least two major problems: The computational power is limited and the simulations would therefore take too long with the present infrastructure. Secondly, a simulation which is fully realistic would be as complicated to analyze as the real world and an understanding would equally hard to archive (Wunsch, 2010). Conceptualization and simplification is a crucial step to describe and eventually understand the system. Climate models are thus an approximation of the reality, with the target to understand the simulated part of the Earth System and to compute it in a realistic time frame.

However, with a climate model it is possible to constrain the contributions of the different processes, since they can be switched on or off individually. A climate model which simulates a time frame including one or several glacial cycles should consider to have a module of ice sheet dynamics. This is important because the ice age cycle dominates climate variability on those time scales, in particular during the late Pleistocene (Tarasov and Peltier, 1997).

The IceBern2D model follows the previously mentioned track of the simplification but in reverse order, from the simple to a more complex case. The implementation is applied to benchmarks from Huybrechts et al. (1996) to understand the ice flow and prove the results.

The surface mass balance (SMB) does only depend on temperature and precipitation, therefore, it can easily be understood and tested with external forcing. The model gets with each iteration more complex and the results get more realistic. Section 5 gives an outlook where further improvements are welcome and additional processes could even enhance the current results.

A New Model Component: IceBern2D

The model is based on the vertically-integrated ice model of Oerlemans (1981b) and Oerlemans (1982). It is designed to simulate the large-scale behavior of ice sheets which is relevant from the viewpoint of dynamical climatology. It considers the 2D flow of the ice and the isostatic adjustment which describes the lowering and uplift of the surface induced by the glacial loading. The transport of ice mass, from accumulation to ablation regions, is approximated as a diffusive process, with a positive but variable nonlinear diffusivity. The diffusivity increases with the slope of the ice surface and in particular with the ice thickness. The bedrock sinks or rises to restore the isostatic balance, the ratio of the height above the surface to the depth of the bedrock is $\rho_{bedrock}$:ρ_{ice} which is approximately 3:1 in steady state. This ratio corresponds to the fact, that the bedrock is around three times denser than ice.

The name of the model IceBern2D refers to the existing three dimensional climate model named "Bern3D" which is developed and used at the University of Bern. It is planed to couple these two models unidirectionally in a first stage, at a later stage a two-way coupling is envisaged. The name IceBern2D suggests that the model belongs to the Bern3D model suite. Furthermore, the name also illustrates that ice in two dimensions is the scope of the model from Bern.

Goals and Structure of this Thesis

The goal of this thesis is the implementation and development of a two-dimensional ice sheet model. This includes the formulation of the ice model into code, and its technical implementation of the ice model. Sensitivity studies of the different parameters with respect to the captured ice volume on land are then carried out and discussed.

The thesis is structured in the following way. Section 2 describes the physical base of the model and how it is discretized in time and space. The climate forcings and their

temperature bias are also presented in this section. Furthermore, the different tuning parameters with their chosen values are introduced.

Section 3 presents the results in different subsections which cover various aspects:
The first subsection (3.1) considers idealized cases and thus represents a laboratory for the ice model. The results of the model are assessed with respect to the most important characteristics of the ice flow. Therefore some benchmarks from the EISMINT project (Huybrechts et al., 1996) are taken against which the model is compared.
The second subsection (3.2) contains a simulation regarding Greenland at present-day conditions. This simulation is a first test at realistic conditions and illustrates the model performance.
The main body of the results is presented in section (3.3) which three ensemble simulations covering the northern hemisphere. These simulations are used for a sensitivity study of the different tuning parameters. The influence of each tuning parameter on the minimum sea level is shown and attributed. Furthermore, results at the minimum sea level are used for analysis how well the ice distribution and volume is represented at the last glacial maximum (LGM) compared with other LGM reconstructions. Finally, some regions show an intriguing oscillation in ice sheet volume under constant forcing. The thesis highlights these regions.

Discussion and conclusions are presented in section 4. The most important points are the oscillating ice sheets, the evolution of the ice sheets with respect to the initial ice-free conditions and the overestimated ice volume.

The last section (5) summarizes the key finding and provides an outlook of the possible improvements and field of applications. All proposed improvements are based on the conclusions in section 4.

2 Description of the Ice Sheet Model

The following section builds the base of the IceBern2D model and explains how the initial ice is retrieved, the flow is calculated, how it is implemented, and how special cases are treated.

2.1 Shallow Ice Approximation Flow Dynamics

The following ice model has the purpose to understand the two horizontal dimensional flow in an ice sheet at a high temporal and spatial resolution under different climatic conditions. It is based on the conservation of mass and simulates the ice flow in two dimensions, the vertical flow in the ice sheet is not simulated explicitly. Therefore, the model is not simulating explicitly any internal properties of the ice sheet such as temperature and velocity distributions. The forcing of the model is deliberately chosen to only include precipitation and temperature as forcing, in order to allow for a wide range of applications with coupled and uncoupled climate models, observational data and possibly climate proxy reconstructions. Not all of theses cases provide detailed data on the ice surface energy balance. The current implementation requires these variables to have a daily resolution but less frequent forcing is also possible with small adaptations. Although the model's purpose is to understand the large scale distribution of ice, the resolution of a grid cell is 40x40 km. For the smaller Greenland domain the grid has a finer resolution of 20x20 km. Each grid cell has exactly one vertical layer which stores relevant information such as ice thickness, accumulation, ablation etc.

The basis of the model is the conservation of ice thickness with time as it is shown by Oerlemans (1981b) and Huybrechts et al. (1996):

$$\frac{\partial h}{\partial t} = \underbrace{\nabla \cdot D \nabla Z}_{F} + SMB, \tag{1}$$

where h refers to the ice thickness and $\frac{\partial h}{\partial t}$ to the rate of change of ice thickness. SMB represents the average annual surface mass balance which is described in section 2.3, Z is the elevation of the ice surface above sea level at the grid cell, the sum of bedrock elevation B and ice thickness h. D represents the nonlinear diffusivity after Glen (1955) which is explained in section 2.2. The ice flux F which is explained at a later stage is

represented in this equation as $\nabla \cdot D\nabla Z$. Here, vector differential operator (∇) is two dimensional.

2.2 Ice Diffusivity

The diffusivity used by Huybrechts et al. (1996) builds the base of the IceBern2D model concerning the empirical ice flow. Its origin is the flow law formulated by Glen (1955) which describes the creep of polycrystalline ice:

$$\dot{\epsilon}_{ij} = A\tau^{n-1}\tau_{ij}, \tag{2}$$

where $\dot{\epsilon}_{ij}$ is the three dimensional component of the strain-rate tensor. τ_{ij} corresponds to the stress-deviator tensor which is explained for the two dimensional horizontal case in the next paragraph. The effective stress τ is defined as the square root of the second invariant of the stress-deviator tensor: $\tau = \sqrt{\frac{1}{2}\tau_{ij}\tau_{ij}}$. The flow law exponent n and the rate factor A are two empirical constants. It can be shown that A strongly depends on the ice temperature (Paterson, 1994). Here A is held constant since the temperature distribution is assumed to be uniform with a value close to the melting point. The temperature in a real ice sheet is in general much lower but deformation is concentrated in the area close to the bed which is near the melting point, which justifies a single constant A. Both parameters can be found in table 1. The pre-exponential flow-law parameter A builds a special case, because the value is used to tune the model (see section 2.8) and is varied in the sensitivity study of the model.

For simplification, the model assumes that the ice only deforms by shearing in horizontal planes, the longitudinal stress deviators are neglected. This is the so-called "shallow ice approximation" that can be justified by the fact that the grid spacing is at least a factor ten greater than the vertical extension of the ice. This leads to the application of the flow law in the case of a slab of ice with a constant thickness h as explained by Paterson (1994). The weight of the glacier applies a gravitational pressure downward with $p = \rho_i g h$ which results in a force in z-direction. This first derivation holds only true for an ice sheet of constant thickness in all directions. If we consider a sloped ice surface in x direction, which leads to a gradient of the height above the bedrock h of $\frac{\partial h}{\partial x}$, the stress at the bottom changes. If we additionally note that the vertical shear stresses in the ice sheet are negligible compared to the vertical normal stresses, the ice exerts a shear stress on the bedrock of $\tau_{xz} = -\rho_i g h \frac{\partial h}{\partial x}$. The x in the index of τ_{xz} indicates the direction of the stress,

z is the direction normal to the plane on which the stress is applied. The slab of ice can now be divided into several small xy-planes at a height z above the bedrock. In the following we will consider the one dimensional case only (x direction) with a flat bedrock. This fragmentation leads to the well-known shear stress formula:

$$\tau_{xz}(z) = -\rho_i g(h-z)\frac{\partial h}{\partial x}. \tag{3}$$

The shear stress formula in equation (3) implies that the shear stress decreases with distance from the bedrock. At the surface of the ice sheet the shear stress is zero and reaches its maximum at the bedrock. Equation (3) also shows a dependence of the surface slope: the larger the slope the higher the shear stress.
In the case of our ice model only the horizontal ice flow u in z-direction is of importance, the vertical velocity w in x-direction is much smaller and can be neglected. This leads to a simplification of equation (2). A geometrical approach by Paterson (1994) shows, that the strain rate is approximately half of the u-velocity gradient in z direction. The universal stress deviator τ_{ij} is treated as simple shear in x direction which is identical to the effective stress τ. These assumptions, along with the horizontal shear stress (Eq. 3) lead to the strain-rate tensor (Eq. 4):

$$\begin{aligned}\dot{\epsilon}_{xz} &= \frac{1}{2}\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \simeq \frac{1}{2}\frac{\partial u}{\partial z} = A\tau_{xz}^{n-1}\tau_{xz} \\ &= A\tau_{xz}^{n} = A\left(-\rho_i g\frac{\partial h}{\partial x}\right)^{n}(h-z)^{n}.\end{aligned} \tag{4}$$

To obtain the volume flow in each direction which is of interest for the model, equation (4) needs to be integrated two times. The first time to get the velocity at a specific level above the bedrock z:

$$\begin{aligned}u(z) &= \int_h^z \frac{\partial u}{\partial z}dz \\ &= 2A\left(-\rho_i g\frac{\partial h}{\partial x}\right)^{n}\int_h^z (h-z)^{n}dz \\ &= 2A\left(-\rho_i g\frac{\partial h}{\partial x}\right)^{n}\cdot\frac{1}{n+1}(h-z)^{n+1}\end{aligned} \tag{5}$$

To get the volume flow over the entire vertical extension, the result of equation (5) needs to be integrated over the full height:

$$
\begin{aligned}
M &= \int_0^h u(z)dz \\
&= \frac{2}{n+1}A\left(-\rho_i g\frac{\partial h}{\partial x}\right)^n \int_0^z (h-z)^{n+1}dz \\
&= \frac{2}{n+1}A\left(-\rho_i g\frac{\partial h}{\partial x}\right)^n \left[\frac{1}{n+2}(h-z)^{n+2}\right]_0^h \\
&= \frac{2}{n+1}A\left(-\rho_i g\frac{\partial h}{\partial x}\right)^n \frac{1}{n+2}h^{n+2} \\
&= \underbrace{\frac{2}{(n+1)(n+2)}A(-\rho_i g)^n \cdot h^{n+2} \cdot \left(\frac{\partial h}{\partial x}\right)^{n-1}}_{Diffusivity\ D'} \cdot \frac{\partial h}{\partial x}
\end{aligned}
\tag{6}
$$

This derivation explains the diffusivity in a one dimensional case on a flat bedrock. This is generalized to two dimensions and a relaxing bedrock by Huybrechts et al. (1996):

$$
D = \frac{2A(\rho g)^n}{n+2}h^{n+2}\left[\left(\frac{\partial Z}{\partial x}\right)^2 + \left(\frac{\partial Z}{\partial y}\right)^2\right]^{\frac{(n-1)}{2}} \tag{7}
$$

Here $(n+1)$ is included in the pre-exponential flow-law parameter A and therefore a factor 4 greater than A in equation (6). Due to the introduction of the bedrock relaxation (section 2.4) and bedrock topography the ice thickness h is replaced by the elevation above sea level Z to obtain the true surface gradient. Equation (7) is used in IceBern2D for the diffusion.

Table 1: Values of constants used in the ice model. Note: The parameter A for the exponential flow law is used to tune the model (Table 4) and therefore not constant between different simulations.

		Value	Quantity
n	=	3	Flow-law exponent[1]
A	=	$10^{-16}\ Pa^{-3}a^{-1}$	Pre-exponential flow-law parameter[1]
g	=	$9.81\ m\ s^{-2}$	Acceleration of gravity[1]
ρ	=	$910\ kg\ m^{-3}$	Ice density[1]
A_{ocean}	=	$3.6 \cdot 10^{14} m^2$	Ocean surface[2]
SL_{offset}	=	$7.36\ m$	Sea level offset for an ice free Greenland[3]
Γ	=	$6.5\ K\ km^{-1}$	Temperature lapse rate
		$31{,}536{,}000\ s\ a^{-1}$	Conversion factor for seconds to year

[1] Huybrechts et al. (1996)
[2] IPCC (2007)
[3] Bamber et al. (2013)

2.3 Surface Mass Balance Parameterization

The surface mass balance (SMB) is a crucial value of the ice sheet. It is represented by the difference of the accumulation and ablation

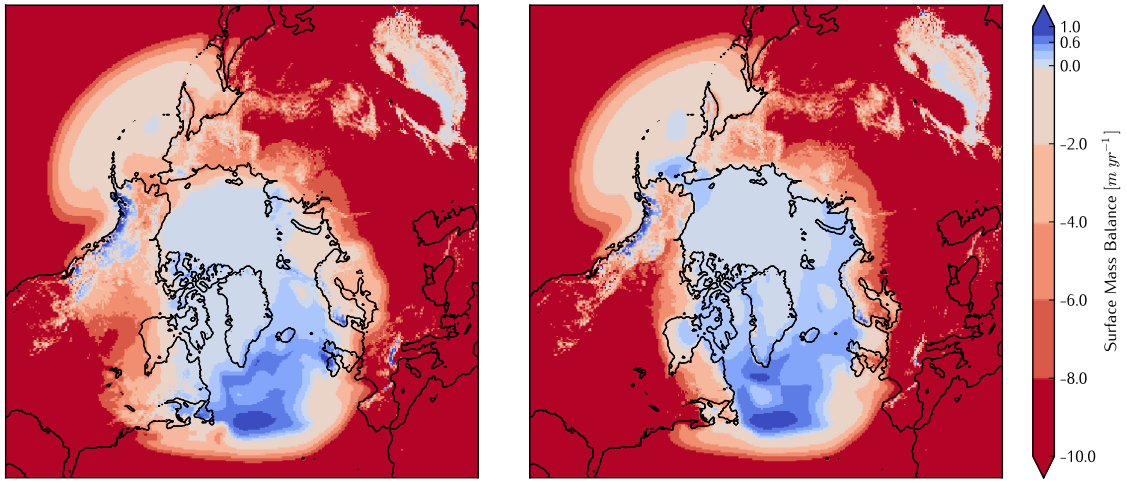
$$SMB = Accumulation - Ablation \tag{8}$$

and determines where the ice sheet gains or loses volume. Positive values lead to a growth, negative values will shrink the ice volume. Accumulation as well as ablation both depend on air temperature. Warm temperatures cause an regime characterized by ablation, while cold temperatures together with precipitation determine the accumulation.
A more sophisticated approach would also include the radiation balance at the ice surface. But this case requires much more information about the atmosphere and increases the complexity of applying the model. The initial purpose of this model was a coupled simulation with the Bern3D model which has a very simplified atmosphere, so the SMB needs to be accomplished with the available data from the Bern3D model.

The data which is used to determine the surface mass balance is an outcome of several CCSM4 climate simulations (Gent et al., 2011; Merz et al., 2013b). Details about the data can be found in the section on the climate forcing (2.7). The ice model has a timestep of one year which makes it impossible to implement a seasonal cycle in the SMB.

The model input temperature is stored as potential temperature with a constant lapse rate Γ of 6.5° $K\ km^{-1}$ and the elevation of the CCSM4 model. Every tenth year the SMB is determined at each grid point. The temperature at each grid point is retrieved from the sum of the product of the constant lapse rate and elevation above sea level Z with the potential temperature T_{pot}: $T = (\Gamma \cdot Z) + T_{pot}$.
Since forcing temperature and precipitation are invariant in time, SMB remains constant as soon as the equilibrium elevation is reached.

The precipitation builds the base for the accumulation. It is assumed that all precipitation at temperatures below the melting point can be accounted as accumulation. Nevertheless, the temperature from the climate data is daily averaged and will scatter around this value during the day. This results in the fact that the ice sheet can also accumulate ice at night when the temperature is below the daily average temperature. And accumulation



a *b*

Figure 1: Surface Mass Balance (SMB) at the beginning of the simulation with $\beta = 8$ mm PDD^{-1} and $A_{temp} = 0°C$. The left figure (a) treats the LGM simulation with LGM topography ("LGMtopo"), the right figure (b) the LGM simulation with present day topography ("PRESENTtopo"). Blue areas have a positive, red areas a negative SMB. The SMB in the Himalaya region does not exceed 0 because the accumulation is set to 0 in this area.

can also take place at day when the temperature is around the freezing point and the daily average temperature much lower. In order to better understand different temperatures of accumulation, A_{temp} is used as a tuning parameter (see section 2.8).

Ablation is parameterized with the sum of days with air temperature above the melting point (positive degree days, PDD) for a given period by a factor called β:

$$Ablation = PDD \cdot \beta. \tag{9}$$

The positive degree day factor β is a simplification to determine the energy which is available to melt the underlying ice or snow. Braithwaite (1995) studied the degree day factor by applying an energy-balance model to Greenland. He shows that β has a high dependency on the different density (ice vs. snow), temperature and albedo of the top layer. The factor for ice is roughly twice the factor of snow and is approximately around 8 $mm\ d^{-1}\ °C^{-1}$. However, in our model there is no distinction between ice and fresh snow, which leads to a mixing between these two factors. The current version does not consider spatial dependence of the factor β. This offers the possibility to use β as a tuning parameter with different values shown in table 4.

The Himalaya is a special case in the ice sheet modeling which is different from the ice

sheet at high latitudes. At high latitudes the glaciers have two characteristic seasons, the winter with a larger accumulation than ablation and summer with a larger ablation. This can be reproduced with the approach of the PDD and the precipitation below the freezing point. However, the Himalaya is different, because it is closer to the equator and already initially at a high altitude. In the Himalaya, the approach used here leads to a very high accumulation and a low ablation rate, which destabilizes the model already in the first timesteps. For this reason, the Himalaya is neglected and the accumulation is set to zero, an investigation to get a realistic SMB in this region would have been too time consuming and beyond the scope of this thesis.

2.4 Bedrock Relaxation Parameterization

Thick ice sheets generate a high mass load on the underlying bedrock. This leads in equilibrium to an isostatic sinking of the bedrock by one third, corresponding to the inverse ratio of rock and ice density: $\frac{\rho_{ice}}{\rho_{rock}} \approx 1/3$. This isostatic process does not take place immediately as it takes some time until an equilibrium is achieved. This time is called the relaxation time τ_{br}.

The isostatic adjustment of the bedrock influences the melting of the ice. If the bedrock yields under the pressure of the ice, the top of the ice sheet sinks to a lower and warmer position. The temperature difference may lead to a melting of the ice at the edge of a stable environment. The smaller ice volume then causes a rebound of the bedrock and the system may start to oscillate. This oscillation can be damped and reach an equilibrium. Alternatively the system may lock into a self-sustained oscillation.

Oerlemans (1981b) provides a possible formulation of the bedrock adjustment. B corresponds to the bedrock elevation, B_0 is the elevation of the bedrock without ice load and h represents the ice thickness. The relaxation time τ_{br} is the time it takes to approach to within 1/e of the equilibrium value. τ_{br} can be used to tune the model and differs in our case between 10,000 and 30,000 years (table 4):

$$\frac{\partial B}{\partial t} = -\tau_{br}^{-1}(\frac{1}{3}h + B - B_0). \tag{10}$$

The approach from equation (10) is a simplified representation of mass flow in the Earth's upper mantle. It only affects the local grid point and no surrounded fields. This approach may be crude but sufficient for the purpose of an ice sheet model of this reduced complexity.

For the elevation of the bedrock without ice load ,B_0, ETOPO1 is used (Amante and Eakins, 2009). ETOPO1 has a resolution of one arc-minute and distinguishes between bedrock and ice surface for Greenland and Antarctica. For our application the resolution is linearly interpolated to an equidistant grid of 40 km in each direction. It is assumed that ice and bedrock are in isostatic equilibrium. The initial position of the bedrock without ice is retrieved by adding one third of the ice thickness to the bedrock elevation. This adaption of the bedrock only affects Greenland, since this is the only place in the chosen domain with an ice sheet.

2.5 Terrain Mask

Due to the fact that most of the grid cells are either water or land without ice, it would be a waste of computational resources to compute the ice flux at these points. To avoid this, a terrain mask with the same size as the domain and three different states is introduced. The first state with the value 0 indicates a grid cell with ice. All water grid cells have the assigned value 1, and the value 2 is assigned to each grid cell characterized by ice free land. At each timestep the terrain mask concerning the land with ice and without ice is updated. The sea level and with it the distribution of ocean grid cells, which involves a more intense calculation with timesteps, is updated every 50 years.

By default all land cells are treated in the terrain mask as ice free land (value 2). As soon as the ice flux or SMB add ice to a grid cell, the mask is changed to the value 0. If the ice retreats at a later stage, the terrain mask is updated.

The terrain mask concerning water is dynamic and depends on the ice volume captured on land. Every grid point with an elevation below the sea level is treated as water (value 1) in the mask. The simulation starts without any ice in the northern hemisphere which leads to an offset in the sea level compared to todays situation. This offset of 7.36 m is equal to the sea level rise if all the ice on Greenland, as major repository in the northern hemisphere, would melt (Table 1) (Bamber et al., 2013). The change of the global mean sea level (GMSL) is retrieved by dividing the water-equivalent ice volume by the ocean area of $3.625 \cdot 10^{14}\ m^2$ (IPCC, 2007). The sea level changes during the simulation and has a large influence on the ice flow, since some shallow bays become land and provide the possibility to distribute ice to a remote area, for example the Baltic Sea.

There are two special cases which are introduced to stabilize the ice flow. If there is a single grid cell of water surrounded by land, this grid cell is converted to land with

the elevation of the current sea level. Otherwise there could be a large ice flow from all surrounding points into this cell which destabilizes the simulation. The second special case appears if the sea level rises at a later stage of the simulation. In this case not only the elevation of the bedrock is considered but also the ice thickness. If the existing ice column with a density of $910\,\mathrm{kg\,m^{-3}}$ is able to displace the water column between the bedrock and sea level, the grid point is still treated as land. As soon as the mass of the water column exceeds the ice mass, the grid cell is converted to a water cell, the remaining ice is lost and contributes to sea level rise.

Before each ice flux calculation, the application checks if one value in the affected cells of the assignment mask is zero. The calculation is only applied if this check holds true. Although this adds overhead to the calculation, it is negligible compared with the computation of the ice flux. The added computational efficiency allows for a larger domain and a higher resolution.

2.6 Discretization

2.6.1 Spatial Discretization

To calculate the ice flow, the equations need to be discretized on a grid. To illustrate how this is implemented in Fortran, selected code lines are presented from the loop over the entire grid. *ix* and *iy* represent the loop counter in both horizontal directions. *dy* and *dx* correspond to the grid spacing and are 40 km in each direction. All constant variables can be found in table 1. The full Fortran source code is available in the appendix in section B.1.

The model is based on a C-grid (Arakawa and Lamb, 1977) as it is presented in figure 2. The staggered C-grid is characterized by a combination of calculated values at the center and the border of the grid. This combination shows the most stable results in the first one-dimensional version of the model. A version with an A-grid, where all quantities are calculated at the center point of the grid cell using centered differences, showed a checkerboard pattern in regions with high ice flux. This pattern was intensified with time until the flux reached almost infinite values. The same phenomenon could also be identified at the first two-dimensional version with an A-grid. A switch to a C-grid resolved this issue.

The elevation gradient ∇Z from equation (1) is calculated at the center of the grid cell

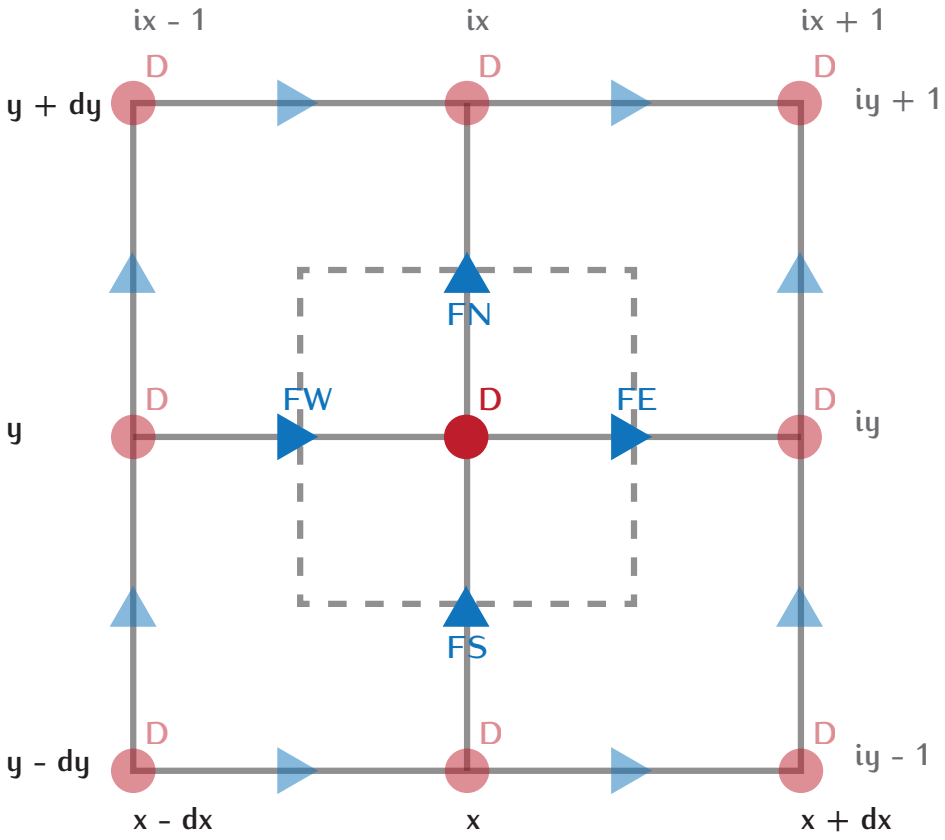


Figure 2: C-grid (Arakawa and Lamb, 1977) adapted to flux and diffusion calculations of the ice model. All solid attributes refer to the center grid point (x|y), the translucent attributes are only an assistance for the rest of the grid. The blue arrows indicate the direction of the flow.

with full steps in both directions. The elevation at the point itself is not involved, only the points in the North and South of the grid cell and the ones in the East and West (centered differences). The elevation includes the elevation of the bedrock plus the overlying ice, if there is any. If a grid cell adjoins a water cell, the sea level is used. There is one simple exception regarding the water elevation: If the the grid cell is an isolated island surrounded by water on all four sides, the elevation of the sea floor is used. Otherwise the elevation gradient would be zero or very close to it and no ice would flow from the island to the water. The island accumulates ice and the elevation increases with each timestep without loosing ice to the neighboring cells. After several thousand simulated years the ice at these islands reaches the maximum value of the variable and the application aborts.

For clarity, the resulting scalar *hgradient* is separated from the diffusion calculation.

```
hgradient = abs( (((elevation(ix+1,iy) &
    - elevation(ix-1,iy))/(2*dx))**2 ) &
    + (((elevation(ix,iy+1) - elevation(ix,iy-1))/(2*dy))**2 ) )
```

The diffusion shown in equation (7) is only dependent on the elevation gradient (**hgradient**) and ice thickness at the grid cell itself. All other values are held constant over the entire simulation.

```
D(ix,iy) = ((2 * A * (rho_ice * g) ** n)/(n + 2)) &
    * (Ice_thickness(ix,iy)**(n + 2)) &
    * (hgradient**((n - 1)/2))
```

At the calculations of the flux, the transition from the center of the C-grid to the border of the cell is realized. The flux between the points is calculated by the interpolated diffusivity multiplied by the elevation gradient. The notation of the flux has its origin in an initial version of the model covering Greenland. In the Greenland domain the flux in the y-direction was the flux in North direction and referred as FN the flux in South direction as FS, respectively. At a later stage, the model was extended to the northern hemisphere with a polar projection and the name of the fluxes were not adjusted. FN and FS are positive in y direction while FW and FE are positive in x direction.
By definition of the grid, the flux FN of a grid cell is identical with the FS flux of the grid cell adjacent to the north: $FN(ix, iy) = FS(ix, iy + 1)$.

Between points with a high elevation difference, the ice flux can reach large values, especially if the sea level changes and one of these grid cells converts from water to land. To ensure numerical stability and to avoid that the flux exceeds the available ice in the source area, the flow is limited to the source ice volume. This limitation ensures that ice is only introduced by the surface mass balance. This check depends on the sign of the flux, therefore, a test in the same code line as the flux calculation is not possible and for clarity not shown here (visible in the Fortran source code in section B.1).

```
FN(ix,iy) = 0.5 * ((D(ix,iy + 1) + D(ix,iy)) * &
    ((elevation(ix,iy + 1) - elevation(ix,iy))/dy ))
```

The new ice thickness is the result of the sum of all fluxes with the previous ice thickness and the surface mass balance (SMB) from section 2.3. All fluxes are divided by the grid resolution (dy and dx) and each term is multiplied by the timestep dt.

```
Ice_thickness(ix,iy) = max(real(0,8), (Ice_thickness(ix,iy) &
    + (((FS(ix,iy) - FN(ix,iy))/dy) * dt) &
    + (((FW(ix,iy) - FE(ix,iy))/dx) * dt) &
    + (surface_mass_balance(ix,iy) * dt)) )
```

With these lines of source code the last step of equation (1) is executed. The new elevation at each grid cell is obtained by adding the ice thickness and calculating the associated bedrock relaxation (section 2.4).

2.6.2 Temporal Discretization

Basic experiments for the northern hemisphere showed that a timestep of one year is appropriate. A smaller timestep of half a year did not improve the results and a doubled timestep sometimes led to large and unstable fluxes.

The Courant–Friedrichs–Lewy criterion (CFL) needs to be satisfied in order to obtain a numerically stable solution:

$$\left| \frac{u\Delta t}{\Delta x} \right| \leq 1. \tag{11}$$

The velocity of the transported characteristic is denoted as u and in the case of the ice model identical with the ice flux F from equation (1). The transposed criterion (Eq. 11) shows, that the flux F has to be smaller or equal than the ratio of the grid size Δx to the timestep Δt:

$$F \overset{!}{\leq} \frac{\Delta x}{\Delta t} = \frac{40\ km}{1\ yr} \tag{12}$$

This CFL condition is satisfied at any time since the ice thickness reaches maximum values around 4 km and the flux does not exceed this value in one year.

Longer intervals are used for processes other than the basic ice dynamics and they are computationally demanding (Table 2). These intervals are based on the simulated years and not on the timestep. The SMB for example is calculated every 10 years and therefore calculated every tenth step with a timestep of one year. If the simulation had a timestep of two years, the SMB would still get calculated every 10 years but now every fifth timestep.

Table 2: Interval of the different calculations. Please note, the chosen timestep is identical to one simulated year. Since the timestep can be varied, the interval changes in respect to the chosen timestep.

Value	Interval
SMB	10 years
Sea level	50 years
NetCDF[4]	1000 years

[4]The interval of the NetCDF output is configurable and can differ from the default value.

The computationally most expensive individual process is the storage of the data into a NetCDF file since the transfer rate to the harddisk is limited and a certain delay is usual. Therefore, the data is not stored at each timestep and an interval is configured. The most important values where stored in the default case every 1000 years which is sufficient for this application.

Bedrock and terrain mask without sea level change are calculated at every timestep.

To quantify the efficiency or demand of the model some metrics may be useful. The model runs on a normal personal computer on one core of an Intel i5-3450 processor of 3.1 GHz. For the northern hemisphere around 800 MB of memory is used and the performance is roughly 133,000 simulated years per hour.

2.7 Climate Forcing

The ice model is forced with climate data from the Community Climate System Model (CCSM4). This dataset is provided from previous studies by Hofer et al. (2012) and Merz et al. (2013b). The two variables of interest are temperature and precipitation at ground level at a daily resolution which are taken as the input to the ice model. Each CCSM4 simulation ran for 33 years, 3 years until the atmospheric system is stable and 30 years which can be regarded as input for the ice model as an averaged seasonal cycle. The ocean is not coupled and is introduced as a boundary condition from a fully coupled CCSM3 simulation of one degree resolution. These CCSM4 simulations have a spatial resolution of $0.9° \times 1.25°$ and a temporal resolution of 30 min. Each stored day is averaged over the effective 30 years to obtain a representative climatology. Table 3 contains the configuration of the different model runs which are used.

2.7.1 Temperature Bias

A comparison of the *PD* CCSM4 simulation with reconstruction data from ERA-Interim (Dee et al., 2011) showed that there is a considerable temperature difference between them. The CCSM3 simulation which is used as ocean forcing for the CCSM4 simulations overestimates the amount of sea ice, for example in the Okhotsk Sea (Collins et al., 2006), which leads to colder temperatures in these areas (Fig. 3). But there are also some areas

Table 3: List of CCSM4 simulations from Hofer et al. (2012) and Merz et al. (2013b) which are used as climate forcing in the ice model. Present-day is noted as PD, Pre-industrial as PI and LGM represents the last glacial maximum. Orbital parameters are calculated according to Berger (1978). SST and sea ice are outputs of corresponding fully-coupled CCSM3 simulations with one degree resolution. Solar forcing is expressed as total solar irradiance (TSI). The topography and ice sheets correspond in the $LGM_{lgmtopo}$ case on LGM ICE-5G mask by Peltier (2004). All simulations have a time resolution of one day and a spatial resolution of one degree which is bi-linearly interpolated to the ice model resolution.

Simulation	Orbital parameters	SST/ sea ice	CO_2 [ppm]	CH_4 [ppb]	N_2O [ppb]	TSI [Wm^{-2}]	Ice sheets/ topography
PD	present	PI 1deg	354	1694	310	1361.8	present
PI	present	PI 1deg	280	760	270	1360.9	present
$LGM_{lgmtopo}$	21ka	21ka 1deg	185	350	200	1360.9	LGM
$LGM_{presenttopo}$	21ka	21ka 1deg	185	350	200	1360.9	present

where the temperature is overestimated. The anomalies range from -12.5 to +5.5°C with an overall mean of -2.985°C.

Three different simulations are introduced to investigate the influence of this temperature bias. The first simulation takes the CCSM4 simulation without any modification as input. The temperature difference is subtracted in the simulation called "bias subtracted" which leads to much warmer temperatures. The simulation named "bias distributed" subtracts the temperature difference and adds the mean of -2.985°C to each grid cell, so that the average has zero bias with the unmodified CCSM4 fields.

The precipitation is not altered in any simulation concerning this temperature bias. But note that the ratio of solid to liquid precipitation of the accumulation is affected by the temperature change.

2.8 Tuning of the Model

To tune the ice model from theoretical considerations to the most realistic known conditions, we adjust a set of variables within their range of uncertainty. In the case of the ice model, there are four different variables which influence the ice flow and distribution.

There are two types of tuning parameters. The first ones change the surface mass balance (SMB) like β and the accumulation temperature A_{temp}. The linear dependence of

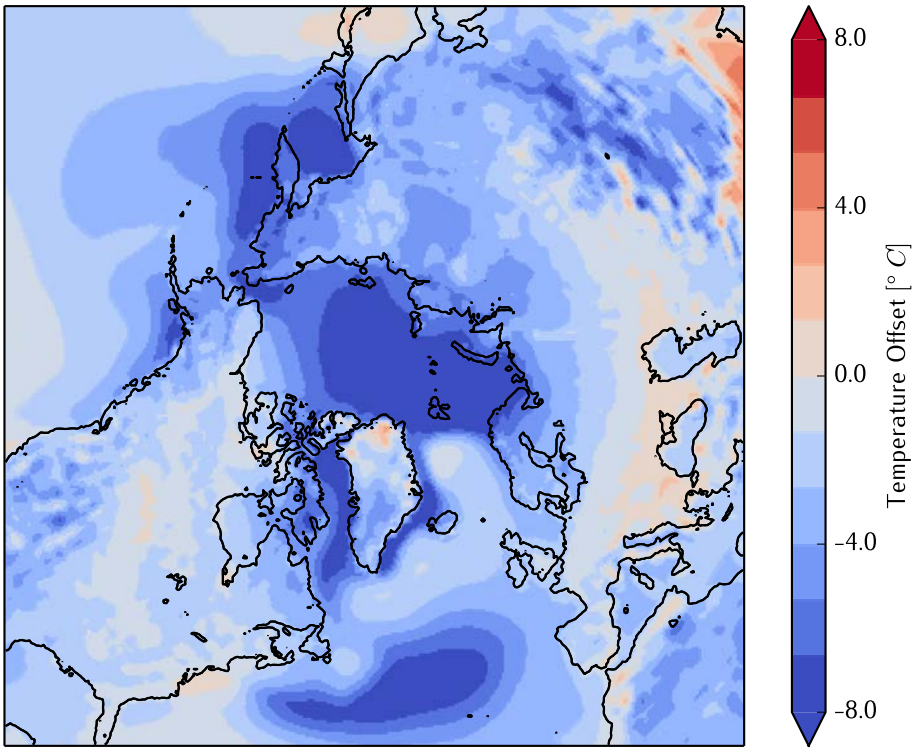


Figure 3: Difference between the CCSM4 and ERA-Interim temperature (CCSM4 - ERA-Interim). Areas in blue have a higher temperature in the ERA-Interim dataset. The mean of the bias over the full domain is -2.985°C.

the accumulation can be found in equation (9). A_{temp} determines at which daily mean temperature the precipitation is treated as accumulation. In both cases, higher values lead to a lower SMB. τ_{br} has an indirect influence on the SMB. The elevation and therefore the temperature decreases at the surface if the elevation yields under the ice. A shorter relaxation time lead to an indirect decrease of the SMB.

The second group influences the ice flow. F_{ice} linearly modifies the pre-exponential flow-law parameter A (Eq. 7). A higher value leads to a faster flow. τ_{br} determines the relaxation time of the bedrock which has an indirect influence on the ice flow. If the bedrock stays longer at its initial elevation, the elevation gradient ∇Z is higher. Otherwise, if the bedrock yields faster under the ice, the ice flow determining elevation gradient (Eq. 7) is smaller and thereby the ice flux smaller.

Table 4: Tuning parameters with their used values. All parameter values together lead to 180 combinations which are investigated for the case of the northern hemisphere with last glacial maximum conditions.

Name	Abbreviation	Unit	Values
Beta	β	$mm \cdot PDD^{-1}$	6; 8; 10
Accumulation temperature	A_{temp}	$°C$	$-2; -1; 0; 1; 2$
Ice flux	F_{ice}	%	75; 100; 125; 150
Bedrock relaxations time	τ_{br}	yr	10, 000; 20, 000; 30, 000

Due to the fact that this thesis covers the initial version of the ice model, the different tuning parameters need to be investigated in their sensitivity and influence of the ice distribution in different areas. Table 4 lists the tuning parameters with their chosen values. Every ensemble consists of all combination of the tuning parameter, which lead to 180 members for each simulation of the temperature bias (section 2.7.1).

3 Results

3.1 Model Validation with Idealized Experiments (EISMINT)

EISMINT (European Ice Sheet Modelling INiTiative) is a series of benchmark experiments to test and compare different ice-sheet models which are based on the same basic principle: conservation of mass, momentum and heat (Huybrechts et al., 1996). To validate our model and their results, two benchmarks are applied. Since the model is based on the same calculation as EISMINT, these experiments have the potential to verify calculations and ensure that the code is free of inadvertent programming errors.

Both EISMINT benchmarks were applied on an artificial quadratic and flat domain to verify the symmetric flow of the ice. The extension in each direction is 1500 km with a consistent grid size of 50 km. This leads to 31x31 = 961 regularly spaced cells. By definition the height of the ice at all boundary grid points (x = 1, x = 31 or y = 1, y = 31) is set to zero at the end of each timestep. An ice discharge to the boundary grid points is still possible but the ice does not accumulate at these grid boxes.
The bedrock at each point is at zero elevation and the bedrock relaxation is turned off. All other constants are identical to the values in table 1.

The two EISMINT tests differ only in the surface mass balance (SMB). In the first experiment, the SMB is a constant accumulation of $0.3\,\mathrm{m\,yr^{-1}}$ over the whole domain ($\mathrm{EISMINT_{fixed}}$). This leads to an ice sheet of square shape which fills the entire model grid. The shape of the ice surface depends solely on the boundary condition and the formulation of the ice flow.

The second experiment takes into account the ice ablation as well and is called "$\mathrm{EISMINT_{freemargin}}$". The SMB is linearly dependent on the distance from the center of the grid and decreases with its distance:

$$\begin{aligned} SMB &= min\{0.5\,\mathrm{m\,yr^{-1}}, s(R_{el} - d)\} \\ d &= \sqrt{(x - x_{summit})^2 + (y - y_{summit})^2}. \end{aligned} \tag{13}$$

The slope s indicates the decrease from the maximum SMB of $0.5\,\mathrm{m\,yr^{-1}}$ by $0.01\,\mathrm{m\,yr^{-1}\,km^{-1}}$. R_{el} is the distance at which the SMB changes from positive to negative values (equilibrium line) and is chosen to be 450 km. However, R_{el} is not identical with the extent of the ice-sheet. Since the ice flows, the ice can also pass the equilibrium line.

The chosen timestep for both tests is 1 yr and the simulation runs for 200,000 years

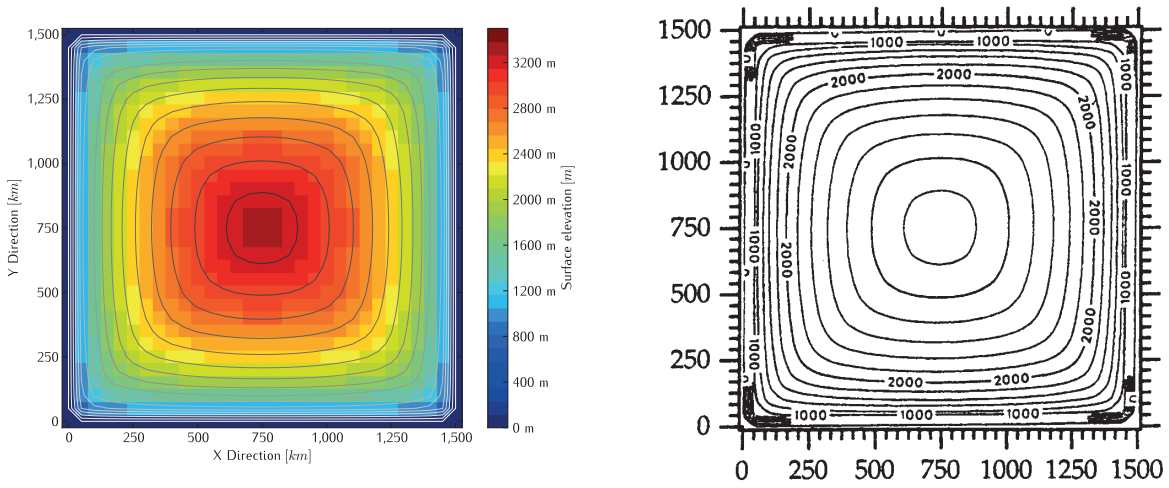


Figure 4: Results of the EISMINT$_{fixed}$ comparison at a steady state around 200,000 simulated years. The image on the left shows ice surface elevation of the IceBern2D model with contour lines of 200 m equidistance. On the right side is the result of the identical test from Huybrechts et al. (1996). Both results have their peak in the center of the grid at an elevation of 3,342.6 m.

although equilibrium is reached already after 25,000 years. The result of EISMINT$_{\text{fixed}}$ and EISMINT$_{\text{freemargin}}$ should be a steady-state ice-sheet configuration which is axis-symmetric.

Our experiment EISMINT$_{\text{fixed}}$ is indistinguishable from the reference by Huybrechts et al. (1996). Both peak in the center of the area are 3,342.6 m above ground (Fig. 4) and therefore our test is in excellent agreement with the original simulations by Huybrechts et al. (1996). EISMINT$_{\text{fixed}}$ shows an axis-symmetric form (Fig. 6) and is also point-symmetric to the center of the grid.

The second benchmark EISMINT$_{\text{freemargin}}$ is not exactly identical to the results from Huybrechts et al. (1996), but very close to it. The peaks are in both cases in the center of the grid and both show an axis-symmetric form. They are point-symmetric to the center of the grid as the EISMINT$_{\text{fixed}}$ results, the axis-symmetric form can be seen in figure 6. However, the elevation of the peak differs slightly by 33.9 m between these simulations (Fig. 5). This may result from the situation that the center of the decreasing SMB is not identical in both simulation (once in the center of the grid cell and once at the border of the cell), since the EISMINT$_{\text{fixed}}$ test showed identical results. Although there was a small difference, the result is satisfactory and no further simulations were carried out to obtain an identical result.

Both EISMINT tests are in very good agreement in respect of the qualitative and quantitative results. The test showed that the source code is free of programming errors concerning the ice flow.

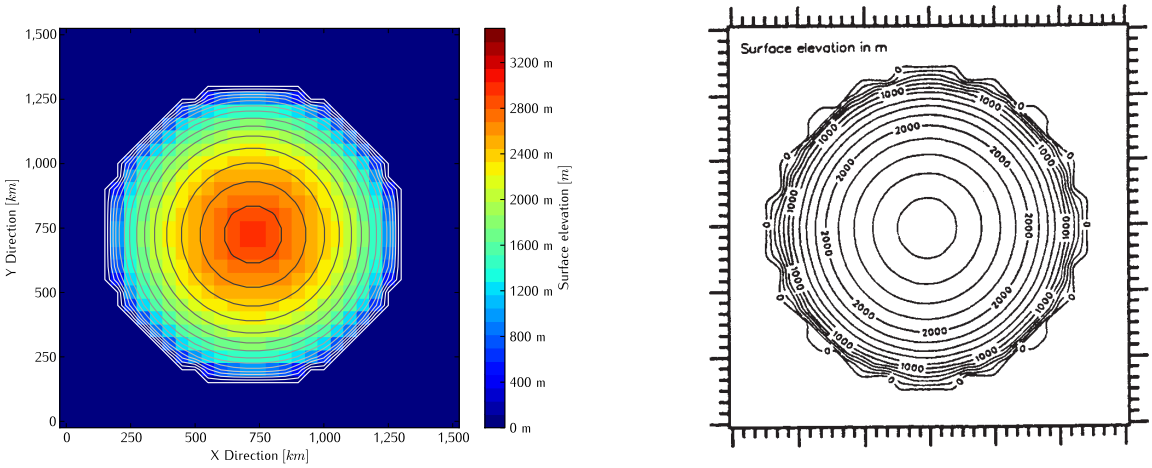


Figure 5: Results of the EISMINT$_{freemargin}$ comparison at a steady state around 200,000 simulated years. The image on the left shows ice surface elevation of the IceBern2D model with contour lines of 200 m equidistance. On the right side is the result of the identical test by Huybrechts et al. (1996). Both results have their peak in the center of the grid, on the left at 2,925.0 m and on the right at 2,958.9 m above ground.

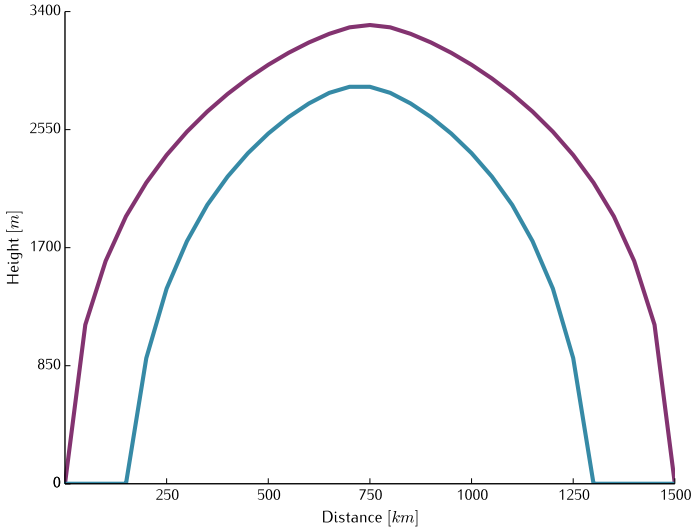


Figure 6: Surface elevation profile along the x-axes through the center of the grid. The profile in purple belongs to the EISMINT$_{fixed}$, the blue one to the EISMINT$_{freemargin}$ test.

3.2 Simulation of the Greenland Ice Sheet

In today's climate, Greenland is the largest volume of ice in the northern hemisphere. The stability of the Greenland ice sheet is of great importance for paleoclimate variations in sea level (Masson-Delmotte et al., 2013; Dahl-Jensen et al., 2013). The Greenland ice sheet could also contribute up to 7 m of additional sea level rise if the threshold for irreversible melt-down is crossed (IPCC, 2013). Therefore, it represents an important test for ice sheet models and will be used here to verify our model in a realistic case. A present-day climate forcing is chosen because it represents a SMB between accumulation and melting around the margins, a balance that is difficult to simulate. In fact, many models simulate an ice sheet that extends all the way to the coast where ice is removed by re-setting its thickness to zero (parameterized calving) instead of melting. Therefore, this simulation also tests the SMB scheme of IceBern2D, an aspect that was neglected in EISMINT.

Only a single simulation is carried out for the Greenland domain, and all conditions are held constant for the simulated 70,000 years. The domain is not really adequate for sensitivity tests of the different parameters, since the domain is too small and the SMB does not differ substantially in this area. The shape of Greenland's bedrock, which is comparable to a bath tub, results in a predominant flow to the center of the island.

The simulation is forced by climate reanalysis data from ERA-Interim (Dee et al., 2011), which has of a spatial resolution of 0.70 degrees with daily mean values from 1980 to 2011. The positive degree day (PDD) factor β is 7 mm PDD^{-1} as proposed by Reeh (1989). The relaxation time τ_{br} of the bedrock is 10,000 yr and the accumulation temperature A_{temp} is 0°C. All other values are identical with the ones from table 1 and are calculated on a grid with 20 km spacing.

Without any parameter adjustments, the result shows already a high similarity with present-day conditions of the ETOPO1 dataset (Figure 7 right). The model underestimates the ice thickness in the center and in the North of the ice sheet. Along the South, East and West borders the ice is thicker compared with present-day estimates. Averaged over the entire domain does the IceBern2D underestimates the ice thickness by 86.7 m or 0.500 m sea level equivalent (SLE). The volume of the Greenland ice sheet based on the ETOPO1 is estimated at $2.83 \cdot 10^{15}\ m^3$ (-7.8 m SLE) compared to $2.67 \cdot 10^{15}\ m^3$ (-7.4 m SLE) from the IceBern2D simulations. A profile along the longitude at the largest extent of Greenland is shown in figure 8.

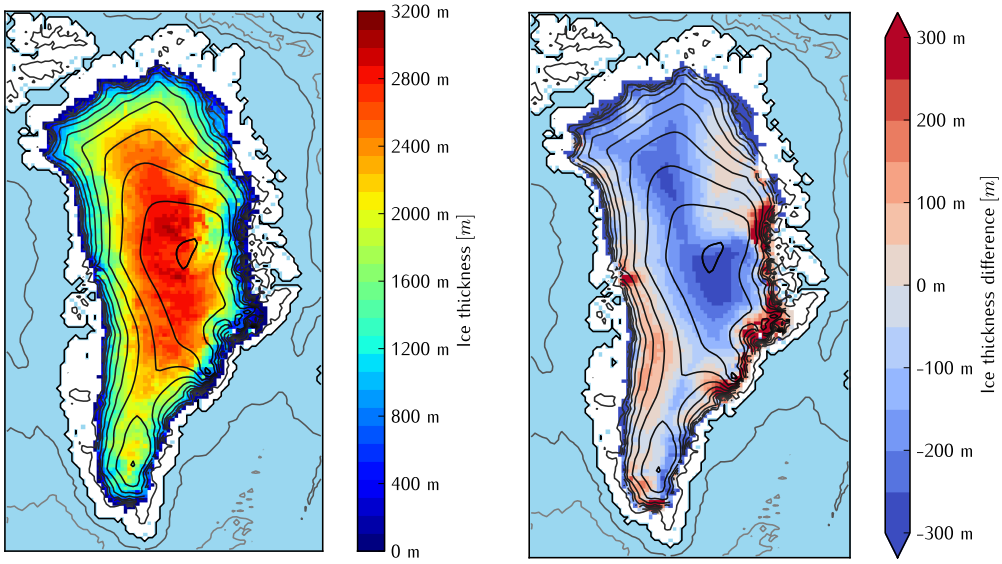


Figure 7: The image on the left shows the ice thickness on Greenland with contour line of 250 m equidistance on ice concerning the elevation. The right image shows the difference between the IceBern2D and ETOPO1 dataset. Overall is the mean difference -86.7 m or 0.5 m sea level equivalent. A North/South profile at the position of the largest extent in longitudinal direction can be seen in figure 8.

3.3 Simulation of Glacial Northern Hemisphere Ice Sheets

During the last glacial maximum (LGM) at around 21,000 years before present most of the additional ice compared to the interglacials is located in the northern hemisphere. Especially the Laurentide ice sheet in the northern part of America was the most prominent glacial ice sheet with a thickness of up to 5000 m (Fig 9). The second area of interest is Europe and western Siberia where the ice flowed from north into southerly direction. The Bering Sea was at this time above the sea level and therefore converted to land. Nevertheless, the area of the Bering Strait was ice free although it is far in the north. The LGM is a good test case to tune the ice model to conditions colder than today. The IceBern2D model is forced with temperature and precipitation data from a CCSM4 simulation which is based on the LGM ICE-5G topography of Peltier (2004). This dataset is based on measurements of a variety of geological and geophysical techniques and processed by a Glacial Isostatic Adjustment (GIA) model. Therefore, ice sheet simulations of the IceBern2D model that employ this forcing are named "LGMtopo".

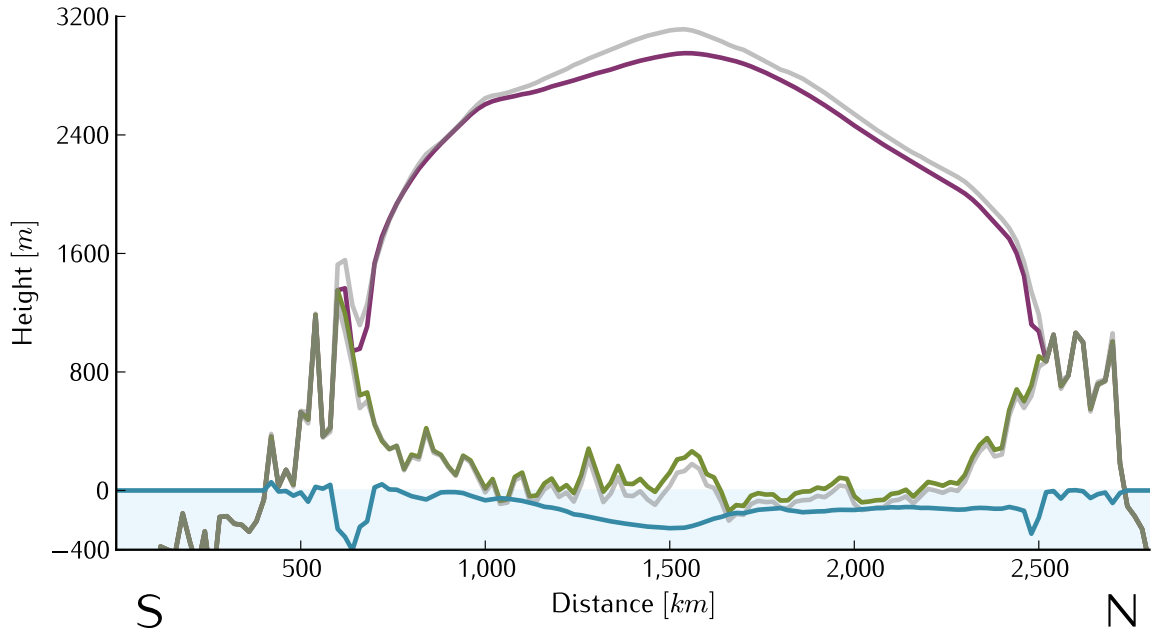


Figure 8: Profile along the x-axis at the position of the largest extend in longitudinal direction. The purple line indicates the surface of the IceBern2D ice sheet, green the bedrock of the simulation. Grey lines represent present-day extent from the ETOPO1 dataset. Blue is the difference along this profile in ice thickness of these two datasets.

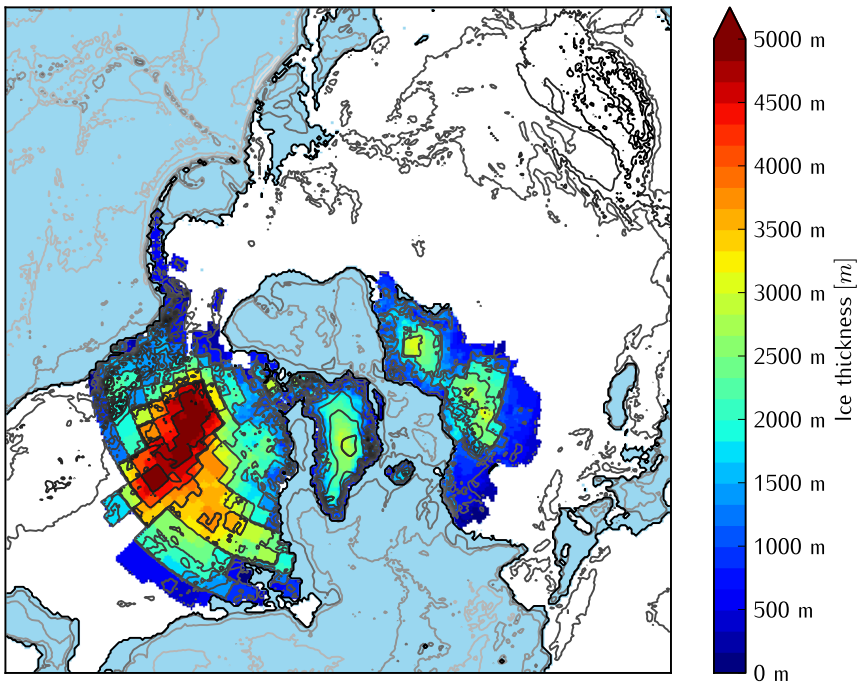


Figure 9: Ice distribution and thickness of ICE-5G (Peltier, 2004). Contour lines at the areas with ice indicate the elevation with 500 m equidistance. Maximum ice thickness is 5223 m. The coast line is consistent with a sea level of -129.3 m.

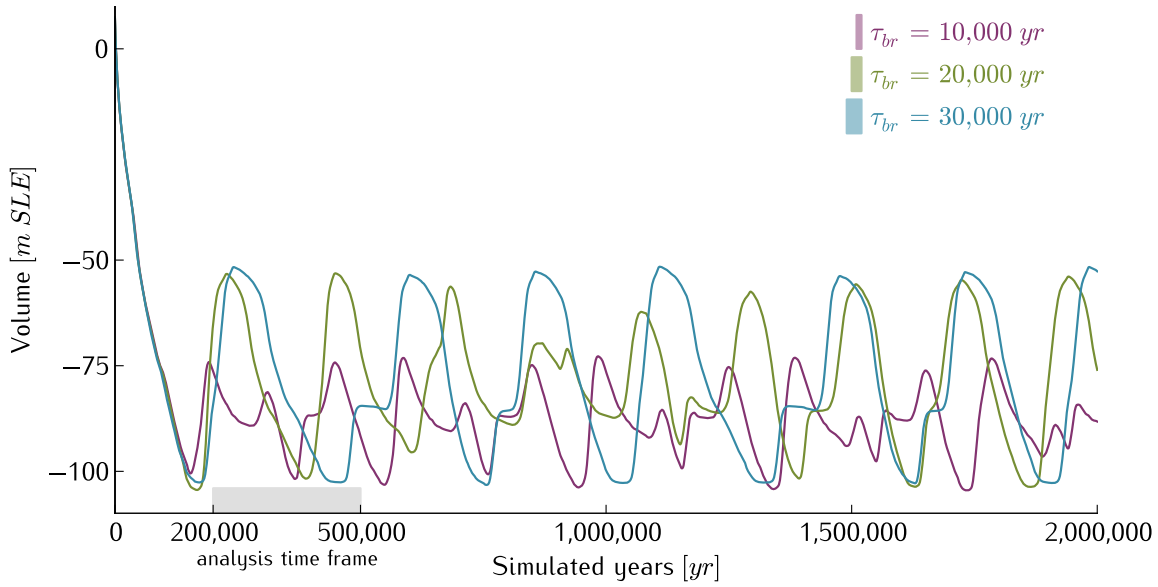
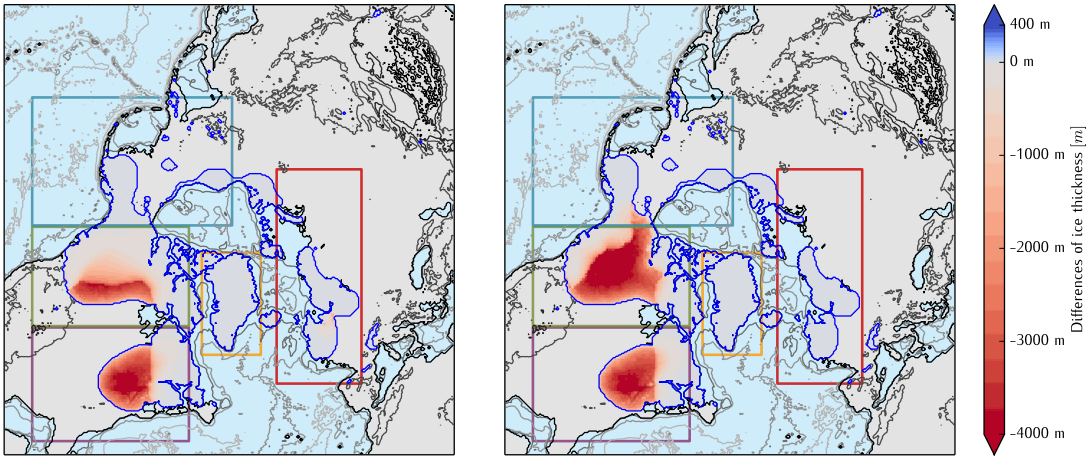


Figure 10: Variation of the sea level with time in respect to different bedrock relaxation τ_{br}. All other parameter are identical between these three simulations: β = 8 mm PDD^{-1}, A_{temp} = 0°C, F_{ice} = 100%. The constant climate forcing is LGMtopo.

3.3.1 Bedrock Oscillation Model

To inform the subsequent ensemble simulations, the ice sheet model was initially run for 2 million years. This long simulation allows us to estimate the time required to reach stable conditions which for a large ensemble should be chosen as short as possible. The runs were executed with three different bedrock relaxation parameters τ_{br} of 10,000, 20,000 and 30,000 years. All other parameters are identical in the 3 experiments and the same as in the simulation of the Greenland ice sheet above, with the exception of β = 8 mm PDD^{-1}. The climate forcing is LGMtopo and constant over the entire time. Every simulation starts with an ice free northern hemisphere and has therefore a positive sea level offset of 7.36 m due to an ice free Greenland at the beginning.

All three simulations show pronounced oscillations in ice sheet volume and the equivalent sea level (Fig. 10) at a constant climate forcing over time. These start after an initial ice build-up that is only weakly influenced by the bedrock relaxation time. We speculate that this spin-up depends mostly on the ice accumulation rate and model parameters related to it. Oscillations of total ice volume are regular, although closer inspection shows that they occur in two main regions including the eastern and western Laurentide ice sheet (Fig. 11 and 12). While the differences between the volume maxima and minima in the eastern part of the Laurentide ice sheet are independent of τ_{br} the western part is influenced by



$\tau_{br} = 10,000\ yr$ $\tau_{br} = 30,000\ yr$

Figure 11: Difference of the ice thickness between the maximum and minimum ice volume for the bedrock relaxation τ_{br} of 10,000 (left) and 30,000 years (right). The maximum sea level for the simulation of τ_{br} = 10,000 years was in the year 984,000 (-72.8 m SLE) and the minimum in the year 1,736,000 (-104.5 m SLE, compare figure 10 & 12). For the τ_{br} = 30,000 years, the maximum corresponds to the year 1,109,000 (-51.6 m SLE) and the minimum to year 757,000 (-103.2 m SLE). Blue contour lines show the maximum spatial extent of the ice sheets. The minimum and maximum of runs with τ_{br} 20,000 years and 30,000 years are very similar, therefore only one of them is shown here. Different areas of interest for analysis of the ice volume change over time are highlighted in colored boxes: Eastern Laurentide in purple, western Laurentide in green, Bering Strait in blue, Greenland in orange and Eurasian ice sheet in red.

the bedrock relaxation. The difference between the maximum and minimum volume for τ_{br} = 10,000 years shows that only at the eastern-most part of the western Laurentide ice sheet participate in the oscillations. With τ_{br} = 30,000 years, large parts of the western Laurentide ice sheet oscillate. In both cases the peak-to-peak difference of the ice thickness is large, a maximum of 4,500 m for τ_{br} = 10,000 years and 5100 m for τ_{br} = 30,000 years, indicating that local deglaciation during the minimum phase is complete.

The total time average ice volume (Fig. 10) is approximately equal in all three simulations. Longer bedrock relaxations times yield larger amplitude variations, because ice can build up higher while the slower bedrock does not yet subside and more ice vanishes once it does yield to the large load. Also, the slower recovery from the depression does not allow ice to re-grow as quickly because the bedrock stays at lower elevations for longer where the SMB is less positive or even negative. Interestingly, this last effect seems to be dominant because the ice volume minimum is very variable in the three simulations while a clear upper limit seems to exist for the maximum (Fig. 10). The phases of the cycle are very

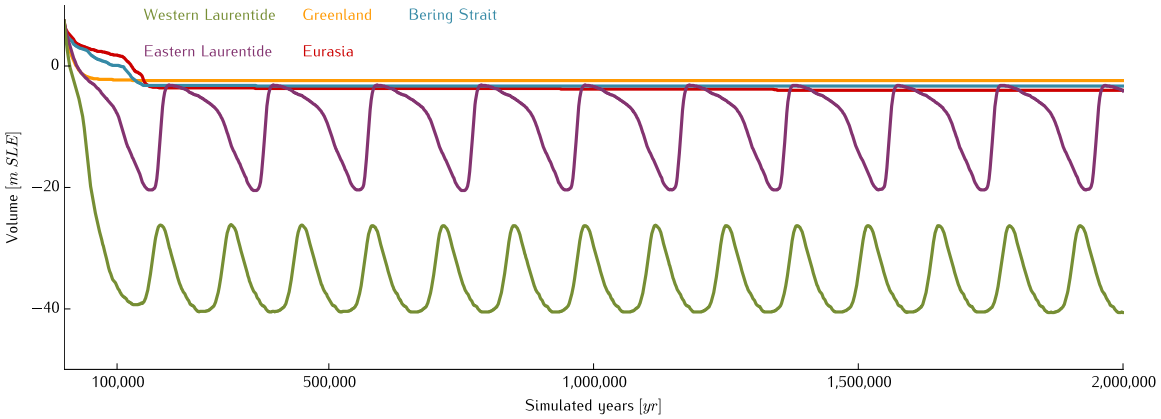
sensitive to the model parameters as Oerlemans (1982) already stated. Each oscillation follows broadly a well defined pattern with minimal variations between cycles.

A decomposition of the northern hemisphere into different areas of interest (colored boxes in figure 11) is done for analysis of the oscillation at a smaller scale. This decomposition helps to understand the superposition of various oscillations that may cause an irregular evolution of the overall sea level (Fig. 10). Areas with oscillations are the western and eastern part of the Laurentide ice sheet (Fig. 11), whereas the other areas are in equilibrium after a build-up of almost 200,000 years (Fig. 12). This general result is independent of bedrock relaxation time τ_{br}. The ice volume in the Bering Strait region shows small amplitude oscillation in phase with the Western Laurentide ice sheet. These do not represent an independent mechanism, but merely stem from the definition of regions (Fig. 11).

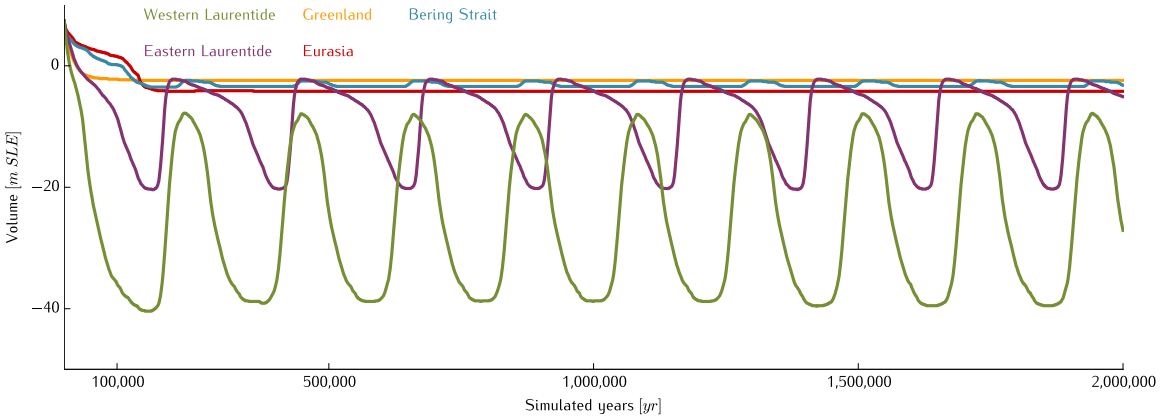
The eastern part of the Laurentide ice sheet shows a similar oscillation for all three bedrock configurations (Fig. 12). The amplitude of roughly 20 meters sea level equivalent is almost the same in all three simulations. However, the frequency changes, from a periodicity of about 200,000 years for τ_{br} = 10,000 years to about 285,000 years for τ_{br} = 30,000 years. The sea level increase , i.e. ice loss, in every simulation is much faster than the buildup of the eastern Laurentide ice sheet.

The impact of the relaxation time is visible in the duration of the minimum sea level. The ice sheet remains longer at the maximum volume with a bedrock relaxation time of 30,000 years compared to τ_{br} = 10,000 years. This is owed to the fact that the bedrock subsides more slowly with a longer relaxation time and therefore the ice sheet preserves its maximum volume for a longer time. The ice loss is very fast in all three cases and a dependence on the bedrock relaxation time is not visible.

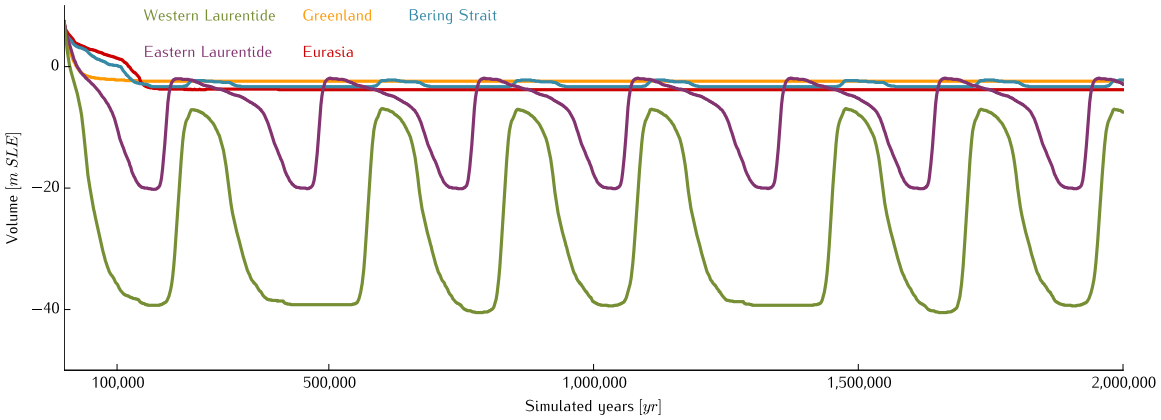
In the build-up process of the ice sheet some changes between the different configurations are visible. Especially at the beginning of the build-up process until the curve experiences a marked acceleration. A reason for the shape of the curve may be the position of the ice sheet. The eastern part of the Laurentide ice sheet is exposed on three sides to a negative surface mass balance (SMB). The ice sheet flows from north to south and is exposed to a negative SMB in the east, west and south (see figure 1). Therefore, the growing process is slow and dominated by a flow from the northern net-accumulation region to the southern net-ablation region until a critical height is reached where surface temperatures fall below the accumulation temperature A_{temp}. The local SMB becomes positive also in the south and accumulation increases the height further with the effect of decreasing temperature at



$\tau_{br} = 10,000\ yr$



$\tau_{br} = 20,000\ yr$



$\tau_{br} = 30,000\ yr$

Figure 12: Volume change in sea level equivalent decomposed into the different areas from figure 11. The sum of the individual lines results in the overall sea level change of figure 10. Eastern Laurentide in purple, western Laurentide in green, Bering Strait in blue, Greenland in orange and Eurasian ice sheet in red.

the surface. A positive ice-elevation feedback is initiated, the ice sheet grows faster and the curve bends. This process depends on the relaxation time because a faster recovery from the depressed state lifts the ice to higher elevation sooner. Equally, it takes longer to reach the point where the SMB turns positive with a longer relaxation time. The area of the ice sheet which is exposed to a positive SMB is smaller at a longer τ_{br} since the bedrock stays longer at lower position where the temperatures are higher. This is also visible in the initial build-up of the eastern Laurentide ice sheet. The bedrock starts at an unstressed position and the build-up process in each simulation is independent of τ_{br} and almost exactly the same.

The western part of the Laurentide ice sheet shows the most prominent changes in the oscillation of the ice volume. Frequency as well as amplitude change with different relaxation time. The amplitude doubles from roughly 15 m in the simulation of τ_{br} = 10,000 years to 30 m in the other simulations with τ_{br} of 20,000 and 30,000 years. The frequency decreases from 14 cycles in 2 million years at a bedrock relaxation of 10,000 years to 7 cycles in 2 million years at a relaxation of 30,000 years. The oscillation at τ_{br} of 20,000 years looks like an magnified version of the simulation with τ_{br} of 10,000 years. However, in the simulation with τ_{br} = 30,000 years changes in the shape of a cycle become evident. While the duration of the sea level high stand is approximately the same in all cycles, the minimum sea level duration is longer than usual on two occasions. The minimum sea level remains stable for around 150,000 years in the long phase while the short phase is similar to the simulations with τ_{br} of 20,000 years. A closer look highlights a possible influence of the eastern Laurentide ice sheet on the minimum sea level of the western Laurentide ice sheet. At the time of the minimum sea level of the western Laurentide ice sheet also the eastern Laurentide ice sheet reduces the sea level by 20 m. These 20 m are critical to unlock new islands in the western part of the Laurentide ice sheet by decreasing the effective depth of ocean channels. The existing ice flows to these islands more easily where the elevation reaches temperatures below the accumulation temperature A_{temp} and ice accumulates. This process generates a very small but evident kink on the long phase of the minimum sea level (Fig. 12).

The same unlocking of islands occurs also when the minimum sea level phase is shorter. The difference here is the timing with respect to phasing of the eastern Laurentide ice sheet. If the latter already accumulated enough ice and therefore depressed sea level sufficiently, the ice bridge to the islands can develop earlier in the cycle of the western Laurentide ice sheet. It is important to observe that the ice that flows northward across the ocean channel is sourced from the same region that later initiates the retreat of the ice. Therefore, the

ice thickness at this point is for a short time thinner in the version where the ice bridge is unlocked later compared to the version where this access opened earlier. It may be that this small difference in ice thickness stabilizes the western Laurentide ice sheet at the time of ice volume maximum in the region where the initial retreat starts. However, a full dynamic understanding would require a more detailed analysis. Despite this incomplete description, the synchronization of the two ice sheets must be by variations in sea level because they are not connected nor does the model include interactions between ice sheet and atmospheric dynamics that could communicate ice volume changes across regions.

The Bering Strait shows a slight dependence on the bedrock relaxation. The simulations with a high western Laurentide amplitude (τ_{br} of 10,000 years and 20,000 years) show a correlation between the western Laurentide and Bering Strait. This is easily explained by the fact that these two regions are difficult to separate and the eastern part of the Bering Strait changes together with the western Laurentide ice sheet (Fig. 11).

Europe and Greenland show a stable ice volume after the first 200,000 years. The SMB for Greenland is positive everywhere already at the initial state at the elevation of the local bedrock (Fig. 1). A decay of the ice volume at a constant climate forcing is therefore unrealistic. Whereas the SMB in Europe has a gradient from positive values in Scandinavia and the northern British Isles to very negative values on continental Europe. The factor that makes the Eurasian ice sheet stable is unknown, there might be an influence of different factors. Mountains may have an influence on the stability of the ice sheet as this is shown by Oerlemans (1981b). And the SMB influence the stability of the ice sheet. This oscillation mechanism operates in the most rapidly changing section of the ice sheet as Birchfield and Grumbine (1985) state. The SMB in Europe may therefore be too small and the European ice sheet does not change fast enough for an oscillation. It is assumed that these two parameters or more participate in the stability of the ice sheet. However, a decomposition of the influence for each parameter is difficult to arrange.

Sea level increase is faster than sea level decrease in all regions (Fig. 12). This is based on the fact that accumulation is limited to the precipitation at a specific grid point and typically does not exceed values of one meter per year. Whereas the ablation can easily melt several meters per year. We do not include calving or other discontinuous mechanical processes that theoretically could lead to even higher ice losses.

We conclude from this analysis that ensemble simulations should run for 500,000 years of which the last 300,000 years will be used for the analysis. The beginning of the analysis is chosen at the first sea level maximum around 210,000 years. Unfortunately

these simulations show, that one entire oscillation cycle is not fully captured within the simulated 500,000 years for a bedrock relaxation τ_{br} of 30,000 years. Nevertheless, these long simulation showed, that a time frame of 300,000 years from 200,000 to 500,000 years (highlighted in figure 10) is adequate and a minimum sea level is realized during this period. The minimum sea level which is a crucial value for the LGM is reached in every simulation in this period and we find that experiments of 500,000 years duration are a reasonable compromise between robustness and computation time tor the full ensemble.

3.3.2 Influence of Tuning Parameters on Sea Level

The different parameters of the model were slightly varied to understand their influence on the ice distribution and extension. For each parameter several values were selected (Table 4) which led to 180 combinations. Each climate forcing (see section 2.7.1) was used for all combinations and builds an ensemble: The LGMtopo forcing which consists of the direct CCSM4 simulation output. The ensemble $\text{LGMtopo}_{\text{bias subtracted}}$ where the temperature bias between the CCSM4 simulations and the reconstructed data ERA-Interim was subtracted. The last ensemble $\text{LGMtopo}_{\text{bias distributed}}$ where the temperature bias is subtracted and the mean of the temperature bias (-2.985°C) is added to each grid cell, so that it has zero bias with the unmodified CCSM4 fields. Every simulation ran for 500,000 years and the state was stored every 1,000 years. Each individual run of these simulations was successful and could be used for later analysis. Overall, roughly 1.6 TB storage was used to save these datasets.

The spread of simulations in terms of sea level is shown for each forcing type and tuning parameter (Fig. 13). With this figure it is possible to see the tendency of the maximum ice volume with variations of the values from each parameter (Table 4). The spreading of the different configuration is visible by the colored boxes denoting 50 percent and 95 percent of the values.

Most of the sea level from the $\text{LGMtopo}_{\text{bias distributed}}$ is below 200 m, while in the $\text{LGMtopo}_{\text{bias subtracted}}$ ensemble it is roughly around 100 m. In every case, the median is above the mean sea level illustrating that the distribution is biased toward smaller ice volumes. This implies that the upper sea levels are grouped closer together or closer to the median compared to the ones below the mean sea level.

It is apparent that the positive degree day factor β has the strongest influence on the locked ice volume in comparison with the other parameters. The mean sea level as well as the the 95 percentile decreases with every 2 mm PDD^{-1} that β increases. This variation

between the different β can also been seen in the other diagrams, where β is shown as column inside each colored box. Generally, the variance decreases with increasing β, only the configuration of 8 mm PDD^{-1} in the LGMtopo$_{\text{bias distributed}}$ ensemble represents an exception.

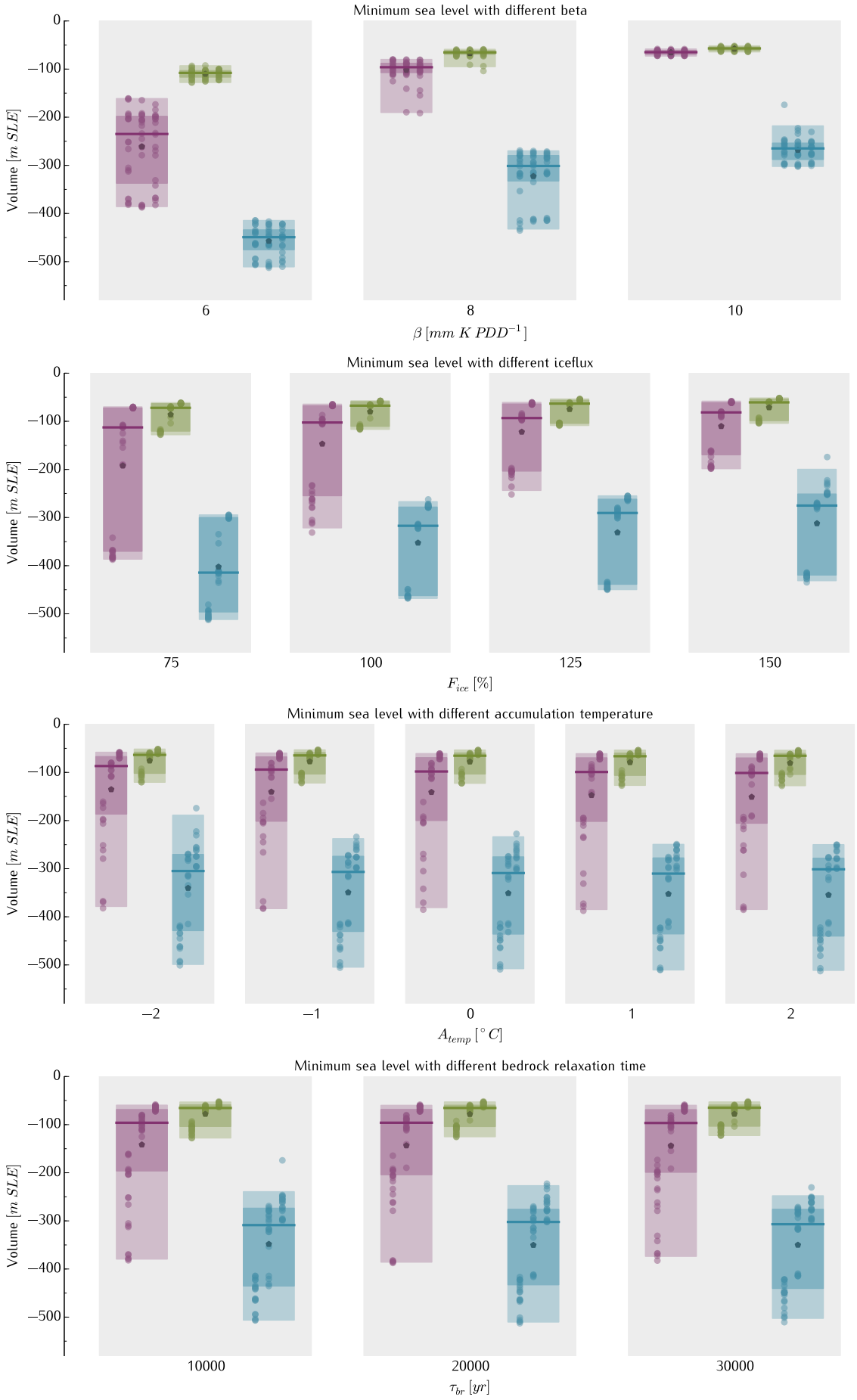
The sea level slightly decreases in every ensemble with increasing ice flux F_{ice}, although the differences in the median are not that obvious in every case. Nevertheless, the mean sea level of each ensemble is moving upwards with higher fluxes. We note the increase of the lower bound in the LGMtopo and LGMtopo$_{\text{bias subtracted}}$ ensemble while the upper limit is fixed. Inside each ensemble, the simulations with the same β are close together, but each of the three β-groups drifts apart from the others. The first and last β-group are close to each edge of the 95 percent box which leads to the result that the 50 and 90 percent box are almost coincide.

The influence of the accumulation temperature A_{temp} on minimum sea level is very small, with increasing A_{temp} resulting in a slightly lower sea level. The influence of A_{temp} compared to the other temperature dependent parameter β is almost zero. The individual influence of these two parameters becomes evident if we compare the two ensembles LGMtopo$_{\text{bias subtracted}}$ and LGMtopo$_{\text{bias distributed}}$. The climate forcing between these two ensembles only differ by a mean temperature of roughly 3°C, the temperature distribution is in both case the same. In every case, the difference of the sea levels between LGMtopo$_{\text{bias subtracted}}$ and LGMtopo$_{\text{bias distributed}}$ is greater than 100 m. But, the LGMtopo$_{\text{bias distributed}}$ simulations with a β of 10 mm PDD^{-1} come not even close to the simulations of LGMtopo$_{\text{bias subtracted}}$ with a β of 6 mm PDD^{-1}. Both ensembles are far away from each other and distinct. The sea level difference between A_{temp} of 3°C, e.g. from -1°C to 2°C, is nearly indistinguishable.

As was shown in figure 10, the different bedrock relaxation time τ_{br} has no influence on the minimum sea level.

With the knowledge of the influence of each tuning parameter (Fig. 13) it is possible to take a closer look at the density distribution of the minimum sea level with respect to

Figure 13 (facing page)*: The dependence of the minimum sea level with respect to the different parameters. The light colored box contains 95 percent of the values, the darker box inside contains half of the values. Median is drawn as line, the average as black pentagon. The different columns in one ensemble represent the different beta configurations, in the beta plot (top right) the bedrock parameter is represented. Different ensembles are represented with different colors: purple = LGMtopo, green = LGMtopo$_{\text{bias subtracted}}$, blue = LGMtopo$_{\text{bias distributed}}$*


Minimum sea level with different beta
Volume [m SLE]
0
−100
−200
−300
−400
−500
6
8
10
β [mm K PDD⁻¹]
Minimum sea level with different iceflux
Volume [m SLE]
0
−100
−200
−300
−400
−500
75
100
125
150
F_ice [%]
Minimum sea level with different accumulation temperature
Volume [m SLE]
0
−100
−200
−300
−400
−500
−2
−1
0
1
2
A_temp [° C]
Minimum sea level with different bedrock relaxation time
Volume [m SLE]
0
−100
−200
−300
−400
−500
10000
20000
30000
τ_br [yr]

the different parameters (Fig. 14). This figure is separated into two parts. The upper part represents a tree plot. At the bottom of the diagram, the minimum sea level for each individual ensemble simulation is shown. Each layer above averages all simulations with equal tuning parameter in order to illustrate the spread they cause. For better readability, the parameters have been ordered so that the one with the greatest influence on minimum ice volume is on top (β) and the least sensitive at the bottom (τ_{br}). With the information about the tendency of the sea level change with respect to the parameter variations (Fig. 13) it is possible to address the individual values (Table 4) at each parameter branch. The lower part of this figure shows a density distribution of the sea level for each climate forcing. It is consistent with the points of the last row in the upper part and distributes these among 100 classes over the whole bandwidth.

The tree plot shows that the influence of the tuning parameter has a clear order. Only a small amount of lines cross each other. Nevertheless, there are a few obvious examples, where the points change the position and join a cluster of another branch. For example, this is the case at the $\text{LGMtopo}_{\text{bias distributed}}$ ensemble where three of four branches from F_{ice} of the intermediate β value tend to a lower sea level. Only one branch has a strong negative trend and joins the group of the lowest β configuration.

The density distribution shows a non-normal distribution for every ensemble of different climate forcing. $\text{LGMtopo}_{\text{bias subtracted}}$ has two obvious groups with a small gap. The group with the upper sea level consists of β configurations with 8 and 10 mm PDD^{-1} (see figure 13), the group with the lower sea level includes all β of 6 mm PDD^{-1}. A similar but not that pronounced constellation is visible in the last $\text{LGMtopo}_{\text{bias distributed}}$ ensemble. The $\text{LGMtopo}_{\text{bias distributed}}$ is only 3°C colder than the $\text{LGMtopo}_{\text{bias subtracted}}$ ensemble which led to a lower sea level and a wider distribution.

3.3.3 Distribution of Ice Volume on Northern Hemisphere

Until now only the minimum sea level, i.e., maximum ice volume, was considered to distinguish the different model outputs. It is unclear how the different ice volumes are distributed over the northern hemisphere. To accomplish this requirement, groups of similar sea levels based on the density distribution were built. The first so called "Group LGM" consists of all members from all ensembles which are in the range of -150 m to -100 m sea level equivalent. This range covers the sea level according to the last glacial maximum which was around -130 m (Clark et al., 2009; Masson-Delmotte et al., 2013). The remaining groups are a result of the clusters in the density plots and are individual for each en-

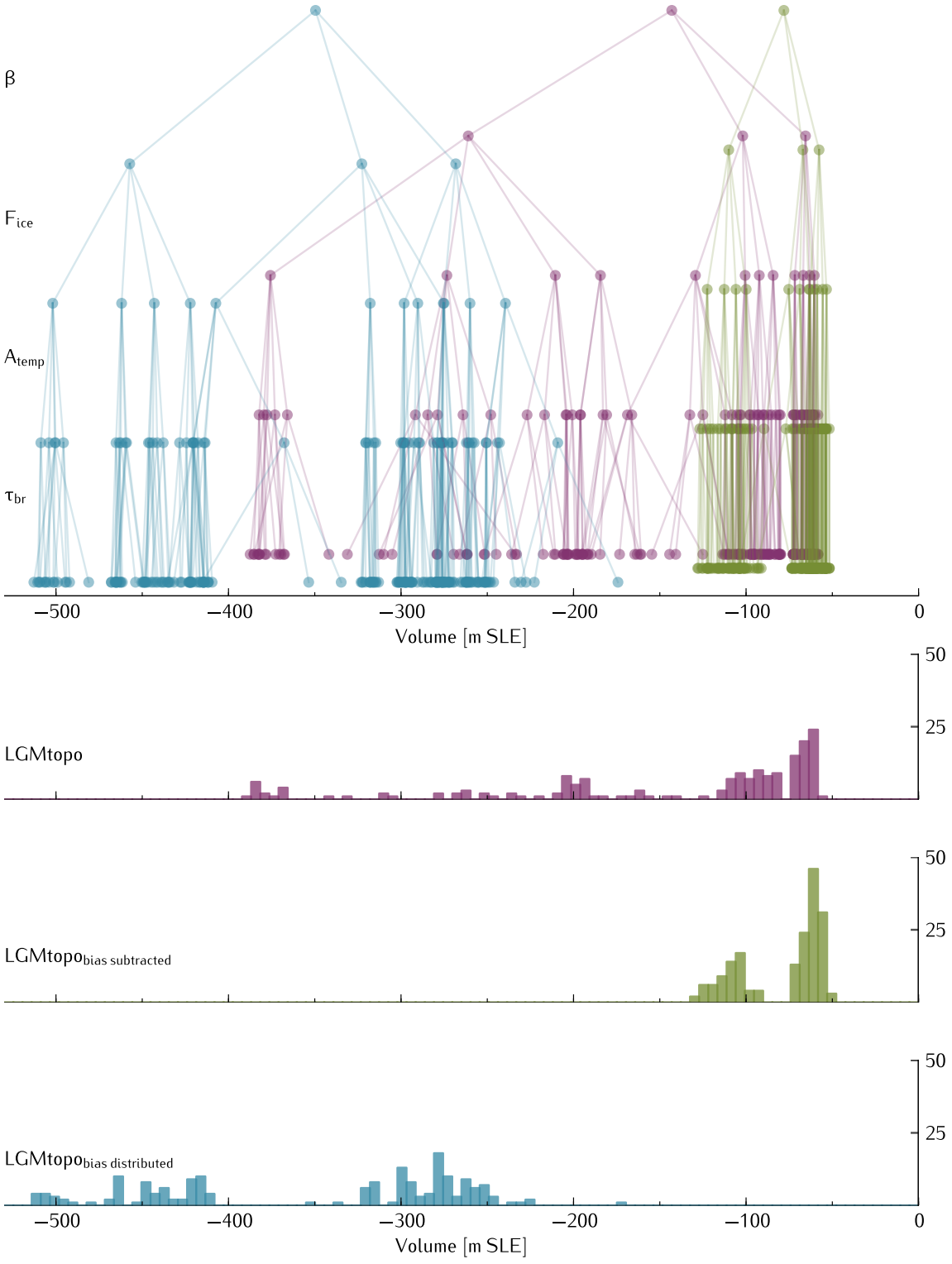


Figure 14: Distribution of the minimum sea level with respect to the influence of each tuning parameter.

Table 5: Group characteristic for different sea levels. All groups are visualized in a density plot in figure 15.

Name	Lower Limit	Upper Limit	Ensemble	# members	⌀sea level
Group LGM	-150 m SLE	-100 m SLE	All ensembles	76	-109.2 m
Group 1	-75 m SLE	-30 m SLE	LGMtopo	61	-65.5 m
Group 2	-120 m SLE	-75 m SLE	LGMtopo	53	-93.1 m
Group 3	-400 m SLE	-120 m SLE	LGMtopo	66	-251.0 m
Group 4	-75 m SLE	-30 m SLE	$\text{LGMtopo}_{\text{bias subtracted}}$	118	-61.5 m
Group 5	-150 m SLE	-75 m SLE	$\text{LGMtopo}_{\text{bias subtracted}}$	62	-107.4 m
Group 6	-370 m SLE	-200 m SLE	$\text{LGMtopo}_{\text{bias distributed}}$	105	-281.0 m
Group 7	-520 m SLE	-400 m SLE	$\text{LGMtopo}_{\text{bias distributed}}$	74	-449.6 m

semble. It has been ensured, that each group has at least 50 members. Table 5 specifies the upper and lower limit and shows the main characteristic of the member quantities and mean sea level of each group. All groups are highlighted in a density plot in figure 15.

The first "Group LGM" (Table 5) is represents by all ensemble members with a sea level between -150 m and -100 m. Nevertheless, only two ensembles (LGMtopo and $\text{LGMtopo}_{\text{bias subtracted}}$) have members with appropriate ice volume. The mean of the ice distribution over the northern hemisphere is shown in figure 16. In North America two distinct domes are simulated from which ice flows towards the Great Plains. But none of the 76 members of group LGM closes the gap between them. A different situation is present at the Bering Sea which is mostly a very shallow sea and therefore drained at a sea level of -65 m. Ice thickness in this area seems to be very thin which is a result of the mean between simulations with and without ice present at this location. This is due to the different temperature distribution in these two ensembles. The ensemble LGMtopo accumulates ice in the Bering Strait while in the $\text{LGMtopo}_{\text{bias subtracted}}$ ensemble it is ice free. The same phenomenon is found over eastern Scandinavia.

The mean of the distributed ice at the time of the minimum sea level of group 1 to 7 can be seen in figure 17. The coast line of each group is different due to the fact that the sea level varies between them (see table 5). Group 4 is omitted because it looks very similar to group 1. Group 7 represents a special case. A very low sea level (-449.6 m) led to a completely ice covered area north of the 55°N. The American, European and Bering Strait sheets are all exhibiting distinct local ice domes.

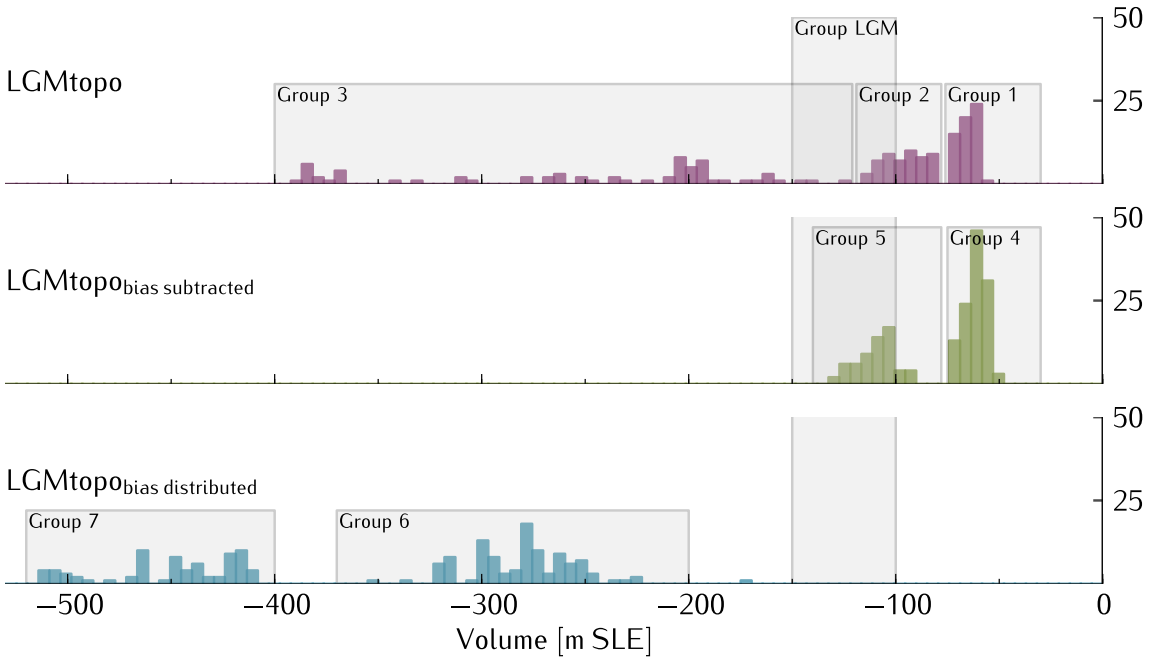


Figure 15: Visualized groups from table 5 and density distribution of the minimum sea level from figure 14. Note that the Group LGM includes all tree ensembles while all the other are limited to one specific ensemble.

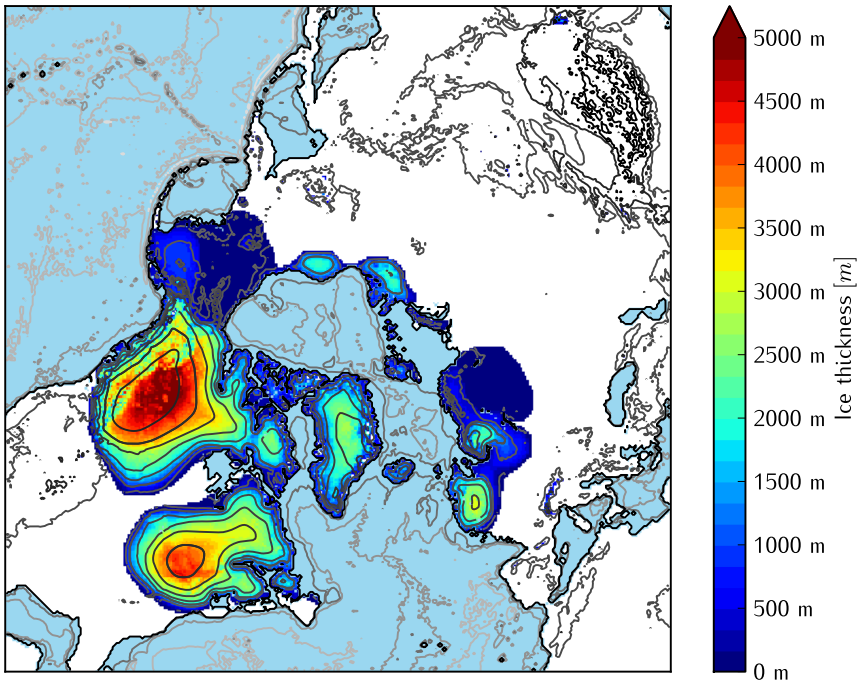


Figure 16: Mean ice thickness of all 76 members of Group LGM (Table 5). Contour lines at the areas with ice indicate the elevation with 500 m equidistance. The coast line is consistent with a sea level of -109.2 m.

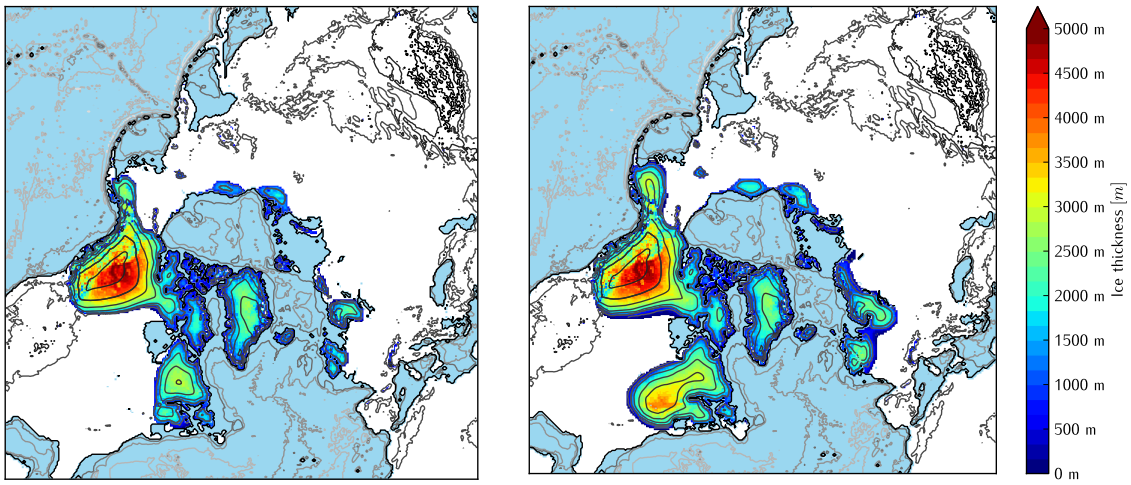
Group 2 and 5 have different climate forcings but were located roughly at the same position of the density distribution of sea level within the range of -130 to -75 m. Although, they differ in the mean sea level by almost 15 m. Two areas are of interest since there are visible differences between these groups. Group 5 shows no ice at the American side of the Bering Strait, while there is a small ice lobe present in group 2. This additional ice from the Bering Strait and the sea level difference seems to be in the two peaks next to the Great Plains, since Europe and the other regions look very similar to each other.

Three main areas contribute most to the total ice volume and are of main interest. The ice sheet in the northern part of America, the Laurentide ice sheet, is the major ice storage. This region is characterized by two flows from West and East into the Great Plains. The Great Plains are either ice free or completely ice covered, only group 3 includes both cases. This contributes to the wide range in ice volume of group 3 which covers more than 200 m of sea level difference, the upper level is comparable with group 5 while the lower level almost reaches group 7.

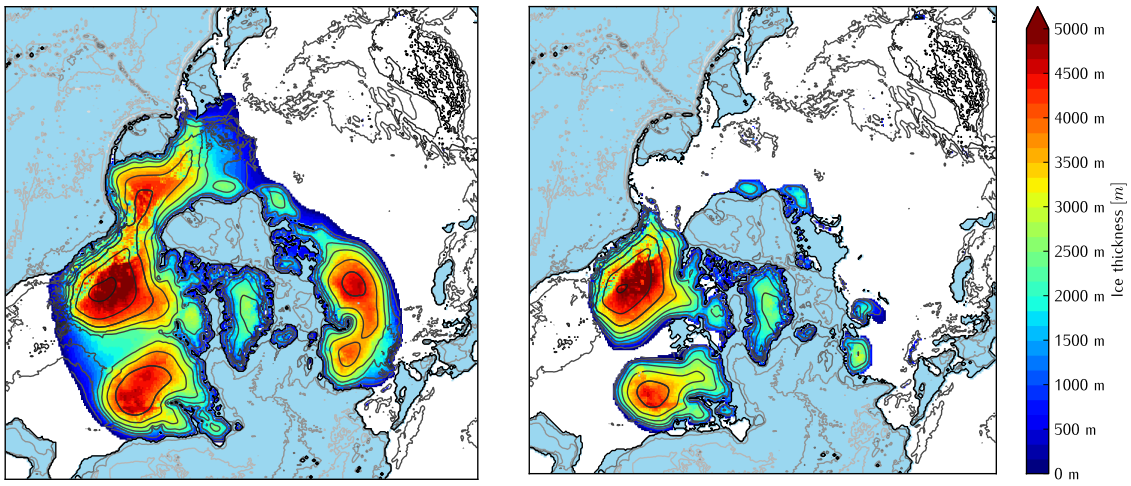
The second area Europe and western Siberia where a clear dependency of the sea level in the extension of the ice sheet into the south is visible. The ice sheet grows from England and Scandinavia in southern direction. The expansion to the south depends more on the minimum sea level than on the different climate of the ensembles. This is also in agreement with the temperature bias between the CCSM4 and ERA-Interim data which is almost neutral in Europe (Fig. 3).

The Bering Strait is the third area of interest because a sinking sea level rapidly exposes the shallow sea floor and allows for ice growth. Although this region falls dry in all groups, not all groups accumulate ice there. A strong dependency of the climate forcing is visible. The first LGMtopo ensemble accumulates ice in this area and in group 3 it flows through the strait passage to Chukotka in far eastern Siberia. The other two climate forcings do not accumulate ice there. The ice flows only in this region if the Laurentide ice sheet is distinct enough (group 6).

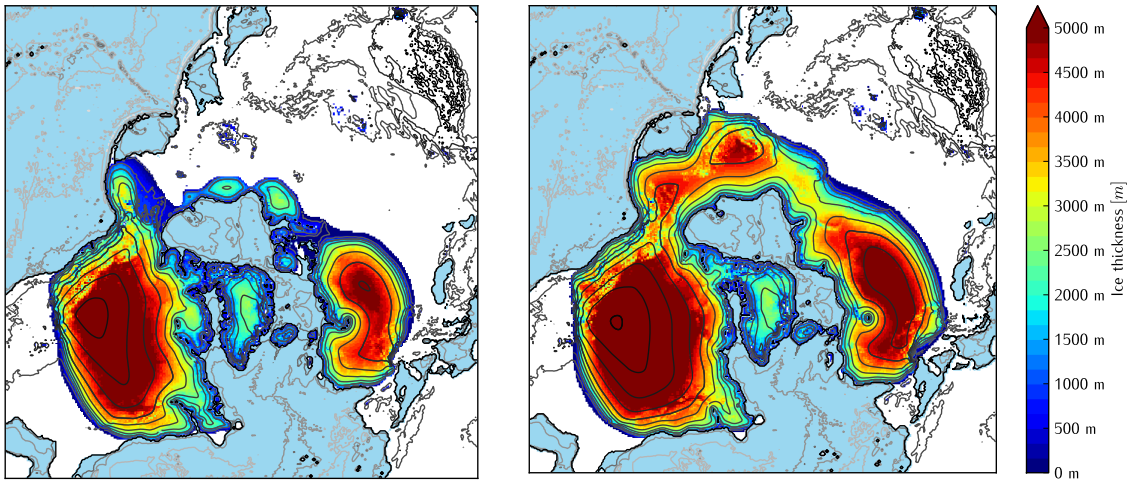
Greenland, the only area in the model domain that caries ice under today's conditions, shows a similar shape in all groups. The peak position and the elevation lines apparently do not change. Even a sea level change of almost 400 m (difference between group 1 and 7) is hardly visible on Greenland at the scale of the northern hemisphere. However, a closer look reveals that there are some differences (Fig. 18). A main difference of all groups compared to the simulations of present-day conditions (Fig. 7) is the fact that the ice reaches the border of the island. This additional ice is responsible for parts of



Group 1 *Group 2*



Group 3 *Group 5*



Group 6 *Group 7*

Figure 17: Ice thickness of the the groups according to table 5. Group 4 is very similar to group 1, this is why it is omitted. Elevation lines with 500 m equidistance at areas with ice, otherwise the equidistance is 2000 m.

the additional sea level equivalent (SLE) compared with present day conditions (Table 6). Another contribution results from the different sea levels. A lower sea level increases the area of Greenland and therefore the island can hold more ice. This is evident from a closer look at Group 1 and Group 6 in figure 18. The ice on Greenland seems to be proportional to the total ice volume on the northern hemisphere (Table 6). However, there is a discrepancy between the IceBern2D simulations and ICE-5G (Peltier, 2004). The groups with a Greenland ice volume comparable to ICE-5G (Group 3 & 6) have an overall sea level which is much lower compared to the one of ICE-5G. Therefore, the relative increase in total ice volume on the northern hemisphere is disproportionately larger than on Greenland in IceBern2D, as compared to ICE-5G.

Table 6: Volume and maximum peak of Greenland in the different datasets. Present day is noted as "PD" while the last glacial maximum as "LGM". The "⌀sea level" refers to the sea level equivalent (SLE) of the full northern hemisphere and is left out for datasets about the present day conditions. Greenland SLE only takes info account ice on Greenland into account.

	Group	⌀sea level	Greenland SLE	Max thickness
PD	ETOPO1[5]		-7.8 m	3304.5 m
	ERA-Interim Simulation[6]		-7.4 m	3051.4 m
LGM	Group LGM	-109.2 m	-9.5 m	3392.2 m
	Group 1	-65.5 m	-9.1 m	3376.1 m
	Group 2	-93.1 m	-9.2 m	3367.0 m
	Group 3	-251.0 m	-10.7 m	3281.6 m
	Group 4	-61.5 m	-9.1 m	3376.5 m
	Group 5	-107.4 m	-9.4 m	3378.4 m
	Group 6	-281.0 m	-10.8 m	3334.6 m
	Group 7	-449.6 m	-13.3 m	3272.8 m
	ICE-5G[7]	-129.3 m	-10.8 m	3559.2 m

The differences of the maximum ice thickness at Greenland between the different LGM groups are not that distinct (Table 6). For an unknown reason the maximum ice thickness for the simulations with a lower sea level is even smaller compared to the ones with a higher sea level. The maximum thickness is roughly 200 m below the value of ICE-5G

[5]Amante and Eakins (2009)
[6]See section 3.2
[7]Peltier (2004)

(Peltier, 2004) for every case. This might be related to a systematic model bias, as the simulation under present day (PD) climate forcing also yields a maximum ice thickness below the data of todays conditions from ETOPO1 Amante and Eakins (2009).

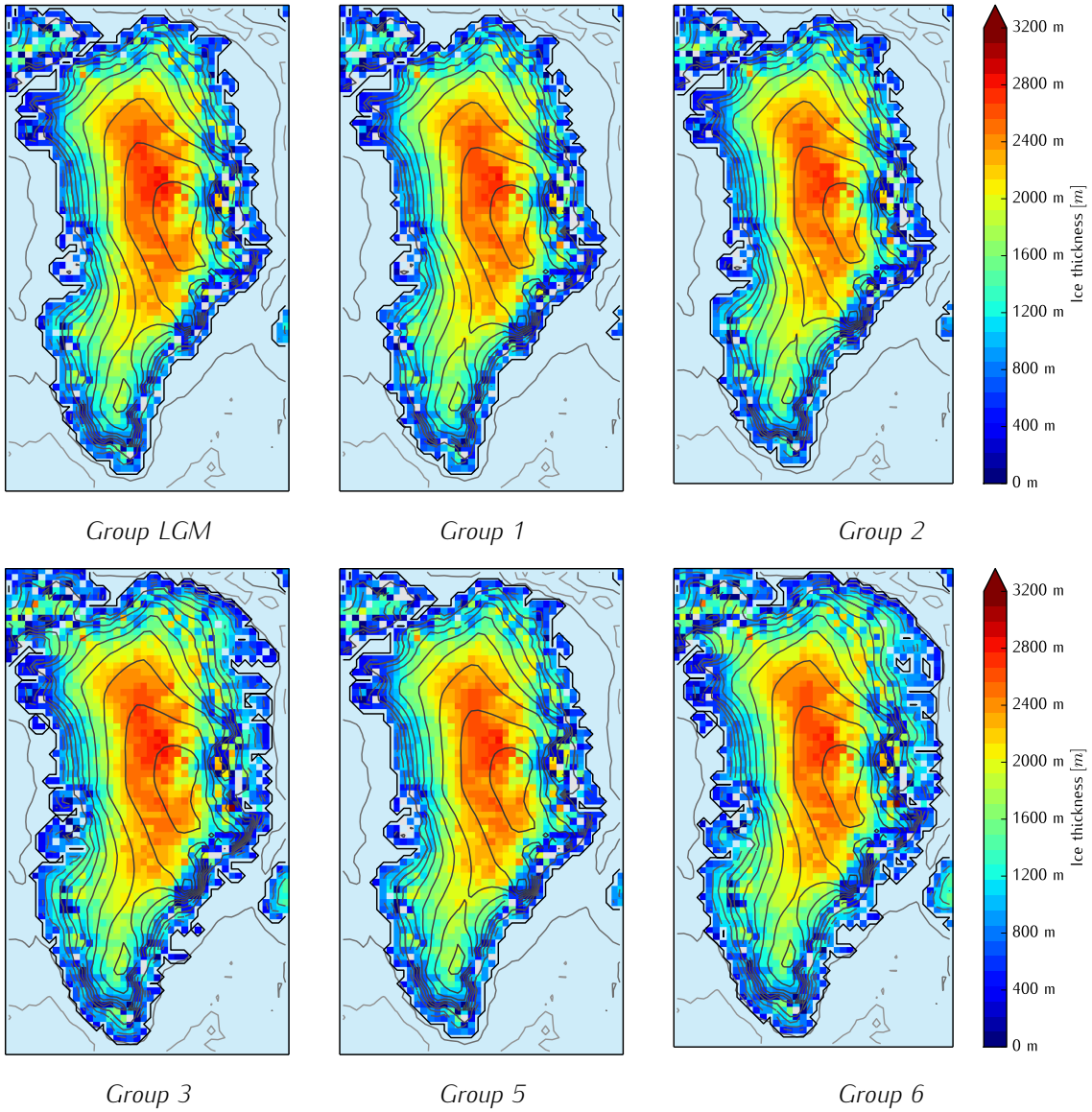


Figure 18: Ice thickness of Greenland for the different groups (Table 5). The sea level is adjusted to the captured ice volume in the northern hemisphere. Contour lines show the elevation with 250 m equidistance and grid spacing is 40 km.

4 Discussion & Conclusions

First experiments in the environment of the EISMINT benchmarks showed a high agreement in their results with Huybrechts et al. (1996). Therefore, the calculations of the ice flow can be considered as robust.

Simulations on Greenland with a present-day climate forcing showed a high similarity with observed ice thickness data of ETOPO1 (Amante and Eakins, 2009). The ice does not reach the coast line at any points (Fig. 7). Therefore, ablation around the margin of the ice sheet balances accumulation further inland, similar to the situation observed in the real world. The simulation starts with an unstressed bedrock, for which its elevation is increased by one third of the observed ice thickness above the observed present-day bedrock. At equilibrium of the ice sheet, the bedrock is close to the observed present-day bedrock (Fig. 8). We conclude that the idealized bedrock relaxation yields sufficiently realistic results.

The change of the sea level with respect of the captured ice masses on land has important influence on the ice distribution. Several areas previously separated by water join and ice can get exchanged. For example Scandinavia connects together with continental Europe and ice build-up in Scandinavia can flow into the south. This would not be possible without a sea level change because the Skagerrak and Kattegat would block the southerly flow.

The choice to ignore the accumulation in the Himalaya area was not significant. Peltier (2004), Tarasov et al. (2012) and others do not detect ice masses in this region that would be relevant for the glacial-interglacial changes considered here.

Oscillating Ice Sheets

The eastern and western part of the Laurentide ice sheet show an oscillation of the ice volume over the entire simulation (Fig. 12) while the other ice sheets are stable over time. This oscillation is induced by the bedrock relaxation only since the climate forcing is constant. Nevertheless, it is not fully understood why the Lauentide ice sheet oscillates and the Eurasian and Greenland ice sheets are in equilibrium. There may be several factors which influence each other.

The most important factor is probably the time which is needed to build up the ice sheet. Both Laurentide ice sheets build up much faster compared to the Greenland and Eurasian

ice sheet (Fig. 12). Therefore, the ice flows at an early stage and relatively fast into the area of a negative surface mass balance (SMB). The bedrock at the Laurentide ice sheet is not able to reach equilibrium in the short time the ice sheet builds up. These areas of rapid change and negative SMB are described by Birchfield and Grumbine (1985) as areas where the oscillation mechanism operates.

The topography may also play an important role. Greenland has the shape of a bath tub: in the beginning all ice flows primarily into the center until a dome in the center is established. A lower bedrock amplifies this shape even more and ice cannot leak until the barrier of the surrounding mountains is reached. The Alps in Europe block the ice flow in south direction. Simulations by Oerlemans (1981b) shows that mountains can stabilize an evolving ice sheet by introducing another nonlinearity. This additional nonlinearity introduces in a certain range of climatic conditions new stable equilibrium solutions. The sloping bedrock of a mountain range also allows an ice-mass discharge to the lower surrounding region (Oerlemans, 1981a). Furthermore, the SMB in Greenland is positive everywhere already at the beginning of the LGM simulations (Fig. 1) the ablation zone is nonexistent and ice is lost by calving only.

Nevertheless, this process is not fully understood and research in this field is rare. Further investigations are needed and may contribute to the understanding of the retreating Laurentide and Eurasian ice sheet at the LGM.

Temperature Bias

One important component of the simulations of the northern hemisphere is the temperature bias in the climate forcing (section 2.7.1). This adaption of the temperature bias changes the ice distribution especially around the Bering Strait (Fig. 17). In these simulations (groups 4 – 6), the ice distribution in this region is similar to the widely used LGM reconstruction of Peltier (2004) (Fig. 9) and Tarasov et al. (2012). Without correction of the temperature bias (group 1 – 3) the temperature is much colder in the Bering Strait and ice accumulates already at an early stage. A similar situation is described by Ziemen et al. (2014) where the missing albedo and moisture blocking in the model is attributed to the accumulation in this region.

Evolution of Northern Hemisphere Ice Sheet

The main focus in the analysis the minimum sea level which is thought to best represent the conditions of the ice sheet maximum and therefore the LGM. Nevertheless, the evolution of the ice sheet is also of interest and can be used to compare the results with other studies. Figure 12 shows the evolution of the different ice sheets in the three runs of 2 million years. In these simulations, the Laurentide ice sheet consists of two independent ice sheets with a similar volume as the ice sheets in group 2 (Fig. 17). The maximum extension of the Eurasian ice sheet and Bering Strait are also similar to group 2. This is reasonable since the climate forcing is identical with the LGMtopo ensemble and β is with 8 mm PDD^{-1} similar to most of the group members (Fig. 14).

The eastern and western parts of the Laurentide ice sheet have the most prominent temporal development in the ice volume. Two individual streams start in the east and west of the North American continent. Simulations with lower temperatures (group 5 & 6) show that these two streams would join each other in the center to form a single ice sheet. Stokes et al. (2012) and Kleman et al. (2013) describe a similar evolution with two convergent ice bodies resulting in the Laurentide ice sheet with a single dome. Nevertheless, the characteristic of the two ice bodies are different. The western volume is more distinct than the eastern volume in the IceBern2D model, while it is the opposite in the simulations of Stokes et al. (2012) and Kleman et al. (2013). This may be a result of the different climate forcings and calculation of the SMB. Stokes et al. (2012) use a forcing which is interpolated between present-day observed climate and LGM climate from Paleoclimate Modelling Intercomparison Project (PMIP) II database (Braconnot et al., 2007) in dependency of a so called "climate weighting index". Kleman et al. (2013) use data from a GCM model scaled with a temperature proxy based on the oxygen-isotope record from a spliced GRIP (Johnsen et al., 1997) and Vostok (Petit et al., 1999) ice core records. The simulations in the IceBern2D model have a constant climate forcing from a simulated atmosphere with a LGM topography. Therefore, the precipitation on the western part of the Laurentide ice sheet is stronger than with an ice-free topography because the ice dome creates a rain shadow for the eastern part.

The development of the Eurasian ice sheet starts in northwestern Europe and evolves to the western Siberia area. Therefore, the LGM Eurasian ice sheet has its origin in a single expanding volume in the IceBern2D model. In contrast, the LGM reconstruction of Kleman et al. (2013) shows an Eurasian ice sheet which evolves from two initially separate European and western Siberian sheets. A temporal reconstruction of the Eurasian ice sheet

during the last glaciation describes a decreasing Barents-Kara ice sheet which feeds the western part of the Eurasian ice sheet at the beginning of the LGM (Svendsen et al., 2004). The SMB decreases at the Siberian part of the ice sheet as soon as the western part of the Eurasian ice sheet grows bigger. This is caused by an isolation of the eastern Eurasian ice sheet by its western part which leads to decreasing precipitation. The proportion between these two ice sheets shifts slowly in western direction. At the time of the LGM is the western part of the Eurasian ice sheet more distinct than the Barents-Kara ice sheet. Such processes are not possible in the IceBern2D model because there is no dynamical atmosphere and the climate forcing is stable over the entire simulation. The amount of precipitation which is accounted for as accumulation changes with the local temperature and therefore elevation changes only. Interactions between different regions do not occur. Nevertheless, a shift of the positive SMB from Siberia in the PRESENTtopo climate forcing to Europe in the LGMtopto climate forcing is visible (Fig. 1). Therefore, a climate forcing which changes from PRESENTtopo to LGMtopo over time could approximate this sequence of the Eurasian ice sheet (see section 5).

Overestimated Ice Thickness

Group 6 of the $\text{LGMtopo}_{\text{bias distributed}}$ ensemble shows a high similarity in the ice distribution compared with ICE-5G (Peltier, 2004). The ice volume on Greenland is virtually identical (-10.8 m) with the reconstruction of ICE-5G. However, ice volumes of the Laurentide and Eurasian ice sheets are by far too large. The mean sea level of group 6 (-281 m) is more than twice the sea level of ICE-5G (-129.3 m). There may be several reasons for this:

The Laurentide ice sheet consist of either one ice body or two ice bodies with a convergent flow in central North America in the IceBern2D. Only in group 3 (Fig. 17), where a large bandwidth of sea levels is covered, both configurations are present. As soon as the western and eastern ice sheets join and a critical height is exceeded, the positive ice-elevation feedback starts. At the initial bedrock level, temperatures in the Great plains are too warm and therefore the SMB is negative (see figure 1). The surface elevation increases dramatically in this area as soon as the two ice bodies join to a single ice sheet. Temperatures are corrected with a constant lapse rate and reach values below the accumulation temperature A_{temp} on at least several days per year. Also, the number of positive degree days (PDD) and with it the ablation become less with a decreasing temperature. Therefore, the SMB changes the sign and gets positive, the elevation increases further. The positive feedback is dominant until all days of the year have temperatures below A_{temp}. At this point, the SMB is weakened by the bedrock sinking under the ice load. It seems that this positive feedback

is too strong as either SMB stays too low to allow the two proto-ice sheets to join, or SMB gets too high when they do join. This suggests a deficient representation of ice sheet-climate interaction that is probably due to the non-adjusting atmospheric dynamics and thus precipitation.

Most investigations for β were based on empirical studies in Greenland and therefore the ice distribution at Greenland, as simulated by the IceBern2D model is very reasonable. Huybrechts et al. (1991) and Braithwaite (1995) propose a positive degree day factor β of 8 mm PDD^{-1} which was varied in the IceBern2D simulations by ± 2 mm PDD^{-1} as a tuning parameter (Table 4). But the relation in total ice volume in the IceBern2D simulations on the northern hemisphere is disproportionately larger than on Greenland compared with other reconstructions such as ICE-5G (see table 6). For understandable reasons no empirical estimations of β for the Laurentide ice sheet exist. The meteorological conditions on Greenland are quite different from the Laurentide ice sheet. For example the cloud coverage on Greenland is relative high compared to other areas in the world especially central Antarctica to which the Laurentide ice sheet is broadly similar. Therefore, a certain rate for a given temperature on Greenland would be representative of a lower shortwave insolation than on the Laurentide ice sheet with a priori uncertain consequences for β.

Furthermore, the PDD factor β is sensitive to the fraction of ice and snow. The PDD factor of ice is roughly twice the factor of snow due to the lower albedo. In the IceBern2D model, the column at a single grid point is represented as homogeneous ice. Therefore, a differentiation between ice and snow is not possible and β which takes into account both conditions is needed. However, it would be worth to carry out some simulations with a lower β between 3 mm PDD^{-1} and 7 mm PDD^{-1} as Reeh (1989) quantifies the PDD factor for snow and ice. The ice sheet may be sensitive to lower β values since the differences of β between 8 mm PDD^{-1} and 10 mm PDD^{-1} are significantly smaller compared to 6 mm PDD^{-1} and 8 mm PDD^{-1} (Fig. 13 & 14). Therefore, the values should be chosen carefully.

Underestimated Elevation at the Summit of the Ice Sheet

Simulations of Greenland under present-day climate conditions as well as LGM simulations of the northern hemisphere underestimate the height of the domes of an ice sheet. The dome in Greenland at present-day conditions is roughly 300 m too low compared to the ETOPO1 dataset (Fig. 7). This offset is also present in the LGM simulations compared to

the ICE-5G dataset (Tab. 6). For the Laurentide ice sheet, these differences are difficult to estimate because no simulation covers both the ice thickness and distribution well enough for a meaningful comparison. Therefore, the maximum ice elevation in the IceBern2D model is not well reproduced and may not be the best parameter to compare with other ice reconstructions.

5 Summary & Outlook

The IceBern2D model is an efficient two-dimensional ice model for paleoclimate studies. The model is well-suited for simulations of ice distributions on a large temporal and spatial scale. Simulations of glacial cycles in the northern hemisphere are potential applications of this model. Calculations of the ice flow field, surface mass balance, bedrock relaxation and sea level change yield reasonable results. Simulations of the ice distribution on the northern hemisphere during the LGM do not yet reproduce the expected pattern but provide an encouraging basis for further work. Small modifications would also provide the possibility of transient simulations. The following list gives an overview of the key findings:

- The ice flow in the IceBern2D model seems to be realistic given the a priori simplifications.
- The parametrized bedrock relaxation scheme is able to reproduce the interaction of ice sheet load with an elastic bed.
- The sea level change is important to link areas separated by shallow water.
- Surface mass balance (SMB) is calculated from temperature and precipitation only and delivers reasonable results in the ice distribution. Nevertheless, some modifications in the SMB are still necessary to come closer to realistic ice volumes.
- The correction of the temperature bias of the CCSM4 data is useful and necessary to simulate a realistic ice distribution on the northern hemisphere.
- Simulations are fast and the computing requirements are low. The simulations can get executed on a simple workstation with common Fortran and NetCDF libraries.
- The IceBern2D code is flexible. Only small modifications are needed to force the simulation with data of another climate model. Some more extensive but still feasible modifications would facilitate an full interactive integration into a climate model.

The climate forcing is kept simple to allow for a wide range of applications, not limited by high requirements on climate forcing data quality. The climate forcing is based on temperature and precipitation at a daily resolution only with an option to use monthly data. Nevertheless, these simplifications also have some disadvantages in the accuracy of the results. Especially the ice ablation would be more precise with full radiation balance data at each grid point. However, simulations of the IceBern2D model give already good

results on the ice distribution at a given climate.

Improvements in surface mass balance

The positive degree day (PDD) factor β results from empirical studies in Greenland (Reeh, 1989; Huybrechts et al., 1991; Braithwaite, 1995). Therefore, this factor captures ablation in Greenland reasonably well which is corroborated by the good representation of the ice sheet in this area. Nevertheless, the PDD factor β is a severe simplification of the surface energy balance that causes the ice to melt. Meteorological factors such as the cloud cover, which influences the insolation and therefore the available energy, are not considered.

A first approach with a β dependent on the latitude may already lead to better results. Another possible factor would be the standard deviation of the daily mean surface temperature which shows large spatial and seasonal differences (Seguinot, 2013). A standard deviation of the daily temperature differences between temperature minimum and maximum could even be more interesting for our application. The daily minimum and maximum temperature is available in the ERA-Interim dataset (Dee et al., 2011). High standard deviations indicate large daily temperature differences and occur rather in winter and distant from the sea. These substantial temperature differences lead to night temperatures below the freezing point. Potential meltwater from the day would therefore refreeze at night. Therefore, β should be lower in regions with a high temperature standard deviation. A similar approach could be used for the accumulation temperature A_{temp} where the standard deviation introduces a weighting for the fraction of the time at which the temperature is below the accumulation point and therefore precipitation counts as accumulation. Another characteristic number could be the difference between daily mean temperature and minimum and maximum temperatures. These two distances starting from the mean temperature indicate if the temperature follows a normalized pattern or if the values cluster in one direction.
All these are considerations how the SMB in different regions could be weighted but they would need to be further elaborated.

An implementation of an available snow-ice surface energy balance model would be the most physically-correct solution. For this additional insolation data is needed for the climate forcing. The benefit of better results would come at the cost of the relative simple climate forcing and therefore a broad field of application.

Precipitation adjustment with temperature

Group 6 of the LGM simulations shows a high similarity with the ice distribution of ICE-5G. Nevertheless, the ice volume is more than twice as big as the ICE-5G volume. A positive ice-elevation feedback contributes to the large ice volumes. At a specific height the temperature is below the accumulation temperature A_{temp} at all days of the year and precipitation of the whole year is accounted for as accumulation. In reality, colder air masses can hold less water vapor. Therefore, the precipitation is usually weaker at higher altitudes or in colder regions. An adjustment of the precipitation in dependence of the temperature and therefore elevation would weaken the positive ice-elevation feedback. The nonlinear relationship between the vapor pressure and temperature known as the Clausius–Clapeyron relation provides a robust physical background and a mathematical description. This precipitation adjustment would lead to thinner ice sheets because the precipitation and therefore accumulation is lower, especially at high altitudes. This effect is also known as "elevation desertification".

Mean temperature bias applied to accumulation only

The ice distribution in group 6 seems to be reasonable as compared to ICE-5G. Nevertheless, the ice thickness in group 6 is way to high, an average of group 5 and 6 would be closer to the reconstruction of ICE-5G. An average between group 5 and 6 would have a similar ice distribution as group 6 with a thinner ice sheet. The density distribution of the minimum sea level (Fig. 14) of this average would cover the spectrum around the LGM sea level of -130 m. The temperature difference of roughly 3°C in the climate forcing at each grid point is the only difference between group 5 ($\text{LGMtopo}_{\text{bias subtracted}}$) and 6 ($\text{LGMtopo}_{\text{bias distributed}}$). There are no spatial and temporal differences in the climate forcing between these two ensembles, only the shift of the mean temperature bias. Therefore, an accumulation identical to the ensemble $\text{LGMtopo}_{\text{bias distributed}}$ and ablation from the ensemble $\text{LGMtopo}_{\text{bias subtracted}}$ could lead to the desired results. This is identical to a shift of the accumulation temperature A_{temp} by 3°C to temperatures of 1°C to 5°C in the $\text{LGMtopo}_{\text{bias subtracted}}$ ensemble.

Climate forcing dependent on sea level

Until now the climate forcing was stable over time and taken from a CCSM4 simulation with LGM topography. Nevertheless, all simulations of the IceBern2D model start with an

ice-free northern hemisphere. Therefore, the initial climate forcing should be precipitation and temperature from an atmosphere simulation with present-day topography. It is well known that large ice sheets have a profound impact on the atmospheric circulation and therefore on precipitation (Li and Battisti, 2008; Pausata et al., 2011). A CCSM4 simulation with LGM orbital parameters and greenhouse gas but present-day topography is available from previous studies (see table 3 and figure 1) but not yet implemented in the IceBern2D. A potential future project could employ a linear relationship between the present-day and LGM topography climate forcing dependency of the sea level for a LGM simulation only. This dependency should start at a sea level offset from Greenland (+7.36 m) with the present-day climate forcing and reach the maximum LGM topography climate forcing at LGM sea level of -130 m. This would change the build-up positions of the ice sheets. Nevertheless, the Laurentide ice sheet would dominate the shift in the climate forcing since the Laurentide ice sheet is the main contributor to the sea level and its change. Therefore, other ice sheets would be connected to the state of the Laurentide ice sheet and their extent could even depend on it.

Such an approach where climate depends on the sea level can also used for a transient simulation with a linear dependency of the present-day and LGM climate forcing. A rising sea level would lead to a more present-day climate forcing while a sinking sea level would induce a LGM climate. The oscillation of the Laurentide could change the climate forcing and some interesting interactions might be expected.

Initialization with ICE-5G

An alternative way to overcome the mismatch between the climate forcing with implicitly-included ice sheets and the ice sheet model initialization without ice is an initialization with the ICE-5G reconstruction. Simulations start already with a maximum ice sheet and therefore a LGM climate forcing with a LGM topography can be used. For the purpose of simulations with a reconstruction of the LGM ice sheet or sensitivity studies a constant forcing can be used. These simulations likely require a shorter spin-up because much of the ice is already present and equilibrium might be reached earlier. An evaluation of the tuning parameters regarding realistic results may be much easier. The simulations start at a reference level, each ice sheet and the overall sea level can be traced for changes. The number of tuning parameter values could even be larger and more tightly sampled without much more time effort in analyzing. Statistical methods, together with suitably defined penalty functions, could be employed to choose the best parameter combination out of the characteristic changes. However, the oscillating ice sheets could challenge such an

evaluation because the sea level may change even if the parameters are close to the initial state. Therefore, a combination of the analysis of the minimum sea level and evolution of the sea level is needed.

Possible simulations and fields of application

Until now all simulations are sensitivity studies of the IceBern2D model. Simulations with specific research questions with respect to past climate have not yet been attempted. Therefore, a short list of possible simulations and their research questions is provided here:

- Transient simulations with a climate forcing following the Milankovitch (1941) cycles. Investigations of potential resonance and synchronization effects between (oscillating) ice sheets and quasi periodic forcing.
- Transient simulations with a climate forcing as simulated for the last 800,000 years, e.g. of the Bern3D climate model.
- Transient simulations with a climate forcing following an internal characteristic, e.g., the area of the ice sheets which influences the albedo of the planet. Therefore, the ablation could be varied depending on the overall ice sheet area. This could highlight internal feedback mechanisms between climate and ice sheets without influences of external changes, e.g. the Milankovitch cycles.
- Paillard (1998) describes a climate system concerning the ice sheet as a multiple-state system with three distinct regimes: interglacial (i), small glaciation (g) and full glaciation (G). A transition between these states is triggered by a specific threshold in ice volume or insolation. An important result of Paillard (1998) is the transition from full glaciation (G) to interglacial (i) which does not persist on the small glaciation (g). A similar cycle is shown by Abe-Ouchi et al. (2013). However, Paillard (1998) does not use a physical model and Abe-Ouchi et al. (2013) can only use interpolated climate data from time-slice GCM simulations, due to computational cost. The IceBern2D model coupled to the Bern3D climate model would be well-suited to complement the earlier findings.

Coupling of the IceBern2D model to the Bern3D climate model

A highlight would be a fully coupled IceBern2D ice module in an existing climate model. The climate model of choice could be the Bern3D model which is developed at the same

institute. The Bern3D model could also benefit from this implementation, because it is used for very long (10^6 yr) simulations for which changes in the cryosphere are important. Ice sheets in the Bern3D model are currently represented as a linear interpolation between present-day and LGM (Peltier, 1994; Ritz et al., 2011). This combination would allow for a wealth of interesting simulations that few modeling groups worldwide have the capabilities to address. Influences of the ocean on the ice sheet and influences of a vanishing Laurentide ice sheet on the ocean currents and ocean biogeochemistry are just some examples. The enigmatic transition from 41-kyr glacial cycles to the so-called 100-kyr world, approximately one million years ago, for which ice sheet-climate interactions were potentially important (Raymo and Nisancioglu, 2003), is another.

A References

Abe-Ouchi, A., Saito, F., Kawamura, K., Raymo, M. E., Okuno, J., Takahashi, K., and Blatter, H. (2013). Insolation-driven 100,000-year glacial cycles and hysteresis of ice-sheet volume. *Nature*, 500:190–193.

Amante, C. and Eakins, B. W. (2009). *ETOPO1 1 arc-minute global relief model: procedures, data sources and analysis*. US Department of Commerce, National Oceanic and Atmospheric Administration, National Environmental Satellite, Data, and Information Service, National Geophysical Data Center, Marine Geology and Geophysics Division.

Arakawa, A. and Lamb, V. R. (1977). Computational design of the basic dynamical processes of the UCLA general circulation model. *Methods in computational physics*, 17:173–265.

Bamber, J., Griggs, J., Hurkmans, R., Dowdeswell, J., Gogineni, S., Howat, I., Mouginot, J., Paden, J., Palmer, S., Rignot, E., et al. (2013). A new bed elevation dataset for Greenland. *The Cryosphere*, 7:499–510.

Berger, A. (1978). Long-term variations of caloric insolation resulting from the Earth's orbital elements. *Quaternary Research*, 9:139–167.

Birchfield, G. E. and Grumbine, R. W. (1985). "Slow" physics of large continental ice sheets and underlying bedrock and its relation to the Pleistocene ice ages. *Journal of Geophysical Research: Solid Earth*, 90:1294–1302.

Braconnot, P., Otto-Bliesner, B., Harrison, S., Joussaume, S., Peterchmitt, J.-Y., Abe-Ouchi, A., Crucifix, M., Driesschaert, E., Fichefet, T., Hewitt, C., et al. (2007). Results of PMIP2 coupled simulations of the Mid-Holocene and Last Glacial Maximum–Part 1: experiments and large-scale features. *Climate of the Past*, 3:261–277.

Braithwaite, R. J. (1995). Positive degree-day factors for ablation on the Greenland ice sheet studied by energy-balance modelling. *Journal of Glaciology*, 41:153–160.

Broccoli, A. and Manabe, S. (1987). The influence of continental ice, atmospheric CO2, and land albedo on the climate of the last glacial maximum. *Climate Dynamics*, 1:87–99.

Cess, R. D., Potter, G. L., Zhang, M.-H., Blanchet, J.-P., Chalita, S., Colman, R., Dazlich, D. A., Genio, A. D. D., Dymnikov, V., Galin, V., Jerrett, D., Keup, E., Lacis, A. A., Le Treut, H., Liang, X.-Z., Mahfouf, J.-F., Mcavaney, B. J., Meleshko, V. P., Mitchell, J. F. B., Morcrette, J.-J., Norris, P. M., Randall, D. A., Rikus, L., Roeckner, E., Royer, J.-F., Schlese, U., Sheinin,

D. A., Slingo, J. M., Sokolov, A. S., Taylor, K. E., Washington, W. M., Wetherald, R. T., and Yagai, I. (1991). Interpretation of Snow-Climate Feedback as Produced by 17 General Circulation Models. *Science*, 253:888–892.

Clark, P. U., Alley, R. B., and Pollard, D. (1999). Northern Hemisphere Ice-Sheet Influences on Global Climate Change. *Science*, 286:1104–1111.

Clark, P. U., Dyke, A. S., Shakun, J. D., Carlson, A. E., Clark, J., Wohlfarth, B., Mitrovica, J. X., Hostetler, S. W., and McCabe, A. M. (2009). The Last Glacial Maximum. *Science*, 325:710–714.

Clark, P. U. and Mix, A. C. (2002). Ice sheets and sea level of the Last Glacial Maximum. *Quaternary Science Reviews*, 21:1–7.

Collins, W. D., Bitz, C. M., Blackmon, M. L., Bonan, G. B., Bretherton, C. S., Carton, J. A., Chang, P., Doney, S. C., Hack, J. J., Henderson, T. B., et al. (2006). The Community Climate System Model version 3 (CCSM3). *Journal of Climate*, 19:2122–2143.

Crucifix, M. and Berger, A. (2002). Simulation of ocean-ice sheet interactions during the last deglaciation. *Paleoceanography*, 17:6–1–6–18.

Dahl-Jensen, D., Albert, M., Aldahan, A., Azuma, N., Balslev-Clausen, D., Baumgartner, M., Berggren, A.-M., Bigler, M., Binder, T., Blunier, T., et al. (2013). Eemian interglacial reconstructed from a Greenland folded ice core. *Nature*, 493:489–494.

Dee, D. P., Uppala, S. M., Simmons, A. J., Berrisford, P., Poli, P., Kobayashi, S., Andrae, U., Balmaseda, M. A., Balsamo, G., Bauer, P., Bechtold, P., Beljaars, A. C. M., van de Berg, L., Bidlot, J., Bormann, N., Delsol, C., Dragani, R., Fuentes, M., Geer, A. J., Haimberger, L., Healy, S. B., Hersbach, H., Hólm, E. V., Isaksen, L., Kållberg, P., Köhler, M., Matricardi, M., McNally, A. P., Monge-Sanz, B. M., Morcrette, J.-J., Park, B.-K., Peubey, C., de Rosnay, P., Tavolato, C., Thépaut, J.-N., and Vitart, F. (2011). The ERA-Interim reanalysis: configuration and performance of the data assimilation system. *Quarterly Journal of the Royal Meteorological Society*, 137:553–597.

Gent, P. R., Danabasoglu, G., Donner, L., Holland, M., Hunke, E., Jayne, S., Lawrence, D., Neale, R., Rasch, P., Vertenstein, M., Worley, P., Yang, Z.-L., and Zhang, M. (2011). The Community Climate System Model version 4. *Journal of Climate*, 24:4973–4991.

Glen, J. W. (1955). The Creep of Polycrystalline Ice. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 228:519–538.

Heinrich, H. (1988). Origin and consequences of cyclic ice rafting in the Northeast Atlantic Ocean during the past 130,000 years. *Quaternary Research*, 29:142–152.

Hofer, D., Raible, C., Merz, N., Dehnert, A., and Kuhlemann, J. (2012). Simulated winter circulation types in the North Atlantic and European region for preindustrial and glacial conditions. *Geophysical Research Letters*, 39.

Huybrechts, P., Letreguilly, A., and Reeh, N. (1991). The Greenland ice sheet and greenhouse warming. *Palaeogeography, Palaeoclimatology, Palaeoecology*, 89:399–412.

Huybrechts, P., Payne, T., EISMINT, I., et al. (1996). The EISMINT benchmarks for testing ice-sheet models. *Annals of Glaciology*, 23:1–12.

IPCC (2007). *Climate Change 2007: The Physical Science Basis. Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*. [Solomon, S., D. Qin, M. Manning, Z. Chen, M. Marquis, K.B. Averyt, M. Tignor and H.L. Miller (eds.)], Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA.

IPCC (2013). *Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. [Stocker, T.F., D. Qin, G.-K. Plattner, M. Tignor, S.K. Allen, J. Boschung, A. Nauels, Y. Xia, V. Bex and P.M. Midgley (eds.)], Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA.

Johnsen, S. J., Clausen, H. B., Dansgaard, W., Gundestrup, N. S., Hammer, C. U., Andersen, U., Andersen, K. K., Hvidberg, C. S., Dahl-Jensen, D., Steffensen, J. P., et al. (1997). The δ18O record along the Greenland Ice Core Project deep ice core and the problem of possible Eemian climatic instability. *Journal of Geophysical Research: Oceans (1978–2012)*, 102:26397–26410.

Kleman, J., Fastook, J., Ebert, K., Nilsson, J., and Caballero, R. (2013). Pre-LGM Northern Hemisphere ice sheet topography. *Climate of the Past*, 9:2365–2378.

Li, C. and Battisti, D. S. (2008). Reduced Atlantic storminess during Last Glacial Maximum: Evidence from a coupled climate model. *Journal of Climate*, 21:3561–3579.

Masson-Delmotte, V., Schulz, M., Abe-Ouchi, A., Beer, J., Ganopolski, A., González Rouco, J. F., Jansen, E., Lambeck, K., Luterbacher, J., Naish, T., Osborn, T., Otto-Bliesner, B., Quinn, T., Ramesh, R., Rojas, M., Shao, X., and Timmermann, A. (2013). *Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth As-*

sessment Report of the Intergovernmental Panel on Climate Change, chapter Information from Paleoclimate Archives. [Stocker, T.F., D. Qin, G.-K. Plattner, M. Tignor, S.K. Allen, J. Boschung, A. Nauels, Y. Xia, V. Bex and P.M. Midgley (eds.)], Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA.

McManus, J. F., Francois, R., Gherardi, J.-M., Keigwin, L. D., and Brown-Leger, S. (2004). Collapse and rapid resumption of Atlantic meridional circulation linked to deglacial climate changes. *Nature*, 428:834–837.

Merz, N., Born, A., Raible, C. C., Fischer, H., and Stocker, T. F. (2013a). Dependence of Eemian Greenland temperature reconstructions on the ice sheet topography. *Climate of the Past Discussions*, 9:6683–6732.

Merz, N., Raible, C. C., Fischer, H., Varma, V., Prange, M., and Stocker, T. F. (2013b). Greenland accumulation and its connection to the large-scale atmospheric circulation in ERA-Interim and paleoclimate simulations. *Climate of the Past*, 9:2433–2450.

Milankovitch, M. (1941). Kanon der Erdbestrahlung und seine Anwendung auf das Eiszeitenproblem. *Acad. R. Serbe*, 132.

Oerlemans, J. (1981a). Modeling of pleistocene European ice sheets: Some experiments with simple mass-balance parameterizations. *Quaternary Research*, 15:77–85.

Oerlemans, J. (1981b). Some basic experiments with a vertically-integrated ice sheet model. *Tellus*, 33:1–11.

Oerlemans, J. (1982). Glacial cycles and ice-sheet modelling. *Climatic Change*, 4:353–374.

Paillard, D. (1998). The timing of Pleistocene glaciations from a simple multiple-state climate model. *Nature*, 391:378–381.

Paillard, D. (2006). What Drives the Ice Age Cycle? *Science*, 313:455–456.

Paterson, W. S. B. (1994). *The Physics of Glaciers*. Butterworth/Heinemann.

Pausata, F. S. R., Li, C., Wettstein, J. J., Kageyama, M., and Nisancioglu, K. H. (2011). The key role of topography in altering North Atlantic atmospheric circulation during the last glacial period. *Climate of the Past*, 7:1089–1101.

Peltier, W. (2004). Global glacial isostasy and the surface of the ice-age Earth: The ICE-5G (VM2) model and GRACE. *Annual Review of Earth and Planetary Sciences*, 32:111–149.

Peltier, W. R. (1994). Ice Age Paleotopography. *Science*, 265:195–201.

Petit, J.-R., Jouzel, J., Raynaud, D., Barkov, N. I., Barnola, J.-M., Basile, I., Bender, M., Chappellaz, J., Davis, M., Delaygue, G., et al. (1999). Climate and atmospheric history of the past 420,000 years from the Vostok ice core, Antarctica. *Nature*, 399:429–436.

Raymo, M. E. and Nisancioglu, K. (2003). The 41 kyr world: Milankovitch's other unsolved mystery. *Paleoceanography*, 18:1011.

Reeh, N. (1989). Parameterization of melt rate and surface temperature on the Greenland ice sheet. *Polarforschung*, 59:113–128.

Ritz, S. P., Stocker, T. F., and Joos, F. (2011). A coupled dynamical ocean–energy balance atmosphere model for paleoclimate studies. *Journal of Climate*, 24:349–375.

Seguinot, J. (2013). Spatial and seasonal effects of temperature variability in a positive degree-day glacier surface mass-balance model. *Journal of Glaciology*, 59:1202–1204.

Siegenthaler, U., Stocker, T. F., Monnin, E., Lüthi, D., Schwander, J., Stauffer, B., Raynaud, D., Barnola, J.-M., Fischer, H., Masson-Delmotte, V., and Jouzel, J. (2005). Stable Carbon Cycle–Climate Relationship During the Late Pleistocene. *Science*, 310:1313–1317.

Stocker, T. F., Timmermann, A., Renold, M., and Timm, O. (2007). Effects of Salt Compensation on the Climate Model Response in Simulations of Large Changes of the Atlantic Meridional Overturning Circulation*. *Journal of Climate*, 20:5912–5928.

Stokes, C. R., Tarasov, L., and Dyke, A. S. (2012). Dynamics of the North American Ice Sheet Complex during its inception and build-up to the Last Glacial Maximum. *Quaternary Science Reviews*, 50:86–104.

Svendsen, J. I., Alexanderson, H., Astakhov, V. I., Demidov, I., Dowdeswell, J. A., Funder, S., Gataullin, V., Henriksen, M., Hjort, C., Houmark-Nielsen, M., et al. (2004). Late Quaternary ice sheet history of northern Eurasia. *Quaternary Science Reviews*, 23:1229–1271.

Tarasov, L., Dyke, A. S., Neal, R. M., and Peltier, W. (2012). A data-calibrated distribution of deglacial chronologies for the North American ice complex from glaciological modeling. *Earth and Planetary Science Letters*, 315:30–40. Sea Level and Ice Sheet Evolution: A PALSEA Special Edition.

Tarasov, L. and Peltier, W. R. (1997). Terminating the 100 kyr ice age cycle. *Journal of Geophysical Research*, 102:21665–21693.

Wunsch, C. (2010). Towards understanding the Paleocean. *Quaternary Science Reviews*, 29:1960–1967.

Ziemen, F. A., Rodehacke, C. B., and Mikolajewicz, U. (2014). Coupled ice sheet–climate modeling under glacial and pre-industrial boundary conditions. *Climate of the Past Discussions*, 10:563–624.

B Appendix

B.1 Fortran Source Code

On the following page can the source code of the IceBern2D model is provided. The code is split into several files with similar functionality. The source code is not 100% identical with the productive version, but the differences have no influence on the functionality and results. The following difference are the most prominent ones:

- Debug information was more detailed in the production version.
- The direction of the ice flow is for historical reasons of the source code in the opposite direction.
- Line breaks are optimized to have a better overview in the limited page width.

The longest lines were split to see the full information. Nevertheless, there are still some lines where not all characters were visible, but the essence of the model should be understandable. If the reader is interested in simulations with the IceBern2D model, the source code may be provided by the author.

The code is not optimized to the last percent of efficiency. It was more important to understand and show the different tasks. Therefore, some calculations are done in several steps where a condensed formulation would increase efficiency.

variables.f90

The file `variables.f90` contains the declaration of all variables and initialization of the constants.

```
1  ! Module with all variables
2  ! Author:        neff@climate.unibe.ch
3  ! Date:          2014/05
4  ! Revision:      1.0b
5  ! SVN Info:      $Id: variables.f90 189 2014-06-01 15:21:53Z basil $
6
7
8  module variables
9
10     !=========================
11     ! Flags
12     !=========================
13     ! Flag, if the values should be written into a netCDF-file
14     logical, parameter :: write_netcdf = .TRUE.
```

```
15      ! Reads the bedrock from a NetCDF file
16      logical, parameter :: read_bedrock = .TRUE.
17      ! Debug Level
18      integer, parameter :: debug = 4
19      ! Copies the netcdf input files to the output directory.
20      logical, parameter :: store_input_netcdf = .FALSE.
21      ! Reads the initial state out of an NetCDF file, which was
            ↪ created with this model.
22      logical, parameter :: read_initial_state = .FALSE.
23      ! Ignore the SMB in the himalaya region.
24      logical, parameter :: ignore_himalaya = .TRUE.
25      ! Enable or disable bedrock
26      logical, parameter :: active_bedrock = .TRUE.
27      ! Adjust the sea level to the ice which is frozen on land. If it
            ↪ is not adjusted, the sea level offset will be 0!
28      logical, parameter :: adjust_sea_level = .TRUE.
29      ! A description of the experiment, which will be used for the
            ↪ folder.
30      character(256), parameter :: experiment_description =
            ↪ 'time5e5_LGM_lgmtopo_adjustsealevel_ParameterDescription'
31
32      ! Unit in the climate netcdf file, which is used to calculate the
            ↪ accumulation
33      ! 1 = m/yr, 2 = m/s, 3 = mm/yr (not yet implemented)
34      integer, parameter :: precipitation_unit = 2
35
36      !=========================
37      ! Constant initialization
38      !=========================
39      ! maximum simulation duration
40      integer, parameter :: maxyears = int(5e5) ! 2e4  = 20'000 years
41      ! seconds in a year (Eismint Table 1): 31556926
42      integer, parameter :: seconds_per_year = 3600*24*365 ! 31556926
43      ! time steps [s]: 10 years = 3600*24*365*10
44      integer, parameter :: dt = int(3600*24*365*real(1))
45      ! distance between two grid points [m] in X-Direction: 20000m =
            ↪ 20km
46      integer, parameter :: dx = 40000 ! 40km
47      ! distance between two grid points [m] in Y-Direction: 20000m =
            ↪ 20km
48      integer, parameter :: dy = 40000 ! 40km
49      ! Diffusivity parameter: Eismint: 10^-16 Pa^-3 a^-1 =
            ↪ 3.1709792e-24 Pa^-3 a^-1 with n = 3
50      real(kind=8), parameter :: A_Eismint = 3.1709792e-24
51      ! Length of domain: Greenland:   1640km = 1.64e6 = 82 * 20 km
52      !                   NH20:       12480km = 1.248e7 = 624 * 20 km
53      !                   NH40:       12480km = 1.248e7 = 312 * 40 km
54      integer, parameter :: L = int(1.248e7)
55      ! Width of domain:  Greenland:   2800km = 2.8e6 = 140 * 20 km
56      !                   NH20:       12480km = 1.248e7 = 624 * 20
57      !                   NH40:       12480km = 1.248e7 = 312 * 40 km
58      integer, parameter :: W = int(1.248e7)
59      ! Exponent in Glens's flow law used in Eismint: 3
60      real(kind=8), parameter :: n_Eismint = 3
61      ! relaxation time for bedrock sinking [s]: 10'000 yr
62      real(kind=8), parameter :: tstar = 3.1536e11
63      ! Acceleration of gravity: 9.81 m s^-2
```

```
64      real(kind=8), parameter :: g = 9.81
65      ! Ice density: 910 kg m^-3
66      real(kind=8), parameter :: rho_ice = 910 ! Eismint, Table 1
67      ! Saturated adiabatic lapse rate
68      real(kind=8), parameter :: lapse_rate = 0.0065 ! degree celsius
            ↪ per m
69      ! Area of the ocean, needed to calculate the sea level rise
70      real(kind=8), parameter :: ocean_area = 3.625e14 ! 3.625e8km^2 =
            ↪ 3.625e14 m^2
71      ! At which position would the sea level be without ice [m]
72      real(kind=8)  :: sea_level_offset = 7.36
73
74      !=========================
75      ! Tuning parameter
76      !=========================
77      ! Flux adjustments to tune the ice flux. This is a percentual
            ↪ adjustment of A_Eismint: 1 = 100%
78      real(kind=8), parameter :: ice_flux_adjustments = 1
79      ! Positive degree days to ablation
80      ! http://www.igsoc.org/journal/59/218/j13J081.pdf
81      ! 3 mm * C-1 * d^-1 = 3.47e-8 m * C^-1 * s^-1 for snow
82      ! 8 mm * C^-1 * d^-1 = 9.26e-8 m * C^-1 * s^-1 for ice
83      ! 3 - 8 mm * C^-1 * d^-1 = 3 / (1000 * 24 * 60 * 60) = 3.5e-8 m *
            ↪ C^-1 * s^-1
84      real(kind=8), parameter :: beta = 7e-8
85      ! accumulation temperature threshhold in Celsius
86      real(kind=8), parameter ::
            ↪ accumulation_daily_temperature_threshold = 0
87
88
89      !=========================
90      ! instance Variable
91      !=========================
92      ! number of grid points in X Direction (Longitude). +1 cause 0 is
            ↪ also a grid box!
93      integer, parameter :: nx = floor(real(W/dx)) + 1 ! declare and
            ↪ init variable, longitude
94      ! number of grid points in Y Direction (Latitude).  +1 cause 0 is
            ↪ also a grid box!
95      integer, parameter :: ny = floor(real(L/dy)) + 1 ! declare and
            ↪ init variable, latitude
96      ! Elevation of the bedrock w/o ice load. This is the relaxed
            ↪ bedrock, the postition where it would be without ice.
97      real(kind=8), dimension(nx,ny) :: Bedrock_Initial = 0
98      ! Elevation of the bedrock, on water the bedrock is equal to the
            ↪ sea level. This is done for internal calculations.
99      real(kind=8), dimension(nx,ny) :: Bedrock = 0
100     ! Ice thickness
101     real(kind=8), dimension(nx,ny) :: ice_thickness = 0
102     ! Elevation of ice surface above sea level
103     real(kind=8), dimension(nx,ny) :: elevation = 0 ! Declare variable
104     ! Distance of each grid box from domain boundary at the bottom,
            ↪ will be multiplied with dy at a later point.
105     real(kind=8), dimension(ny) :: y = (/ (I, I = 0, ny - 1) /) !
            ↪ Declare variable
106     ! Distance of each grid box from left domain boundary, will be
            ↪ multiplied with dx at a later point.
```

```
107     real(kind=8), dimension(nx) :: x = (/ (I, I = 0, nx - 1) /) !
            ↪ Declare variable
108     ! Sea leval
109     real(kind=8) :: sea_level = 0
110
111     ! Some variables needed during runtime
112     integer :: it = 1 ! Loop counter (primary loop, time)
113     integer, parameter :: timesteps =
            ↪ int(real(maxyears)/real(dt)*(3600*24*365))
114     integer, dimension(timesteps) :: myyear = 1     ! Timesteps with
            ↪ the years belonging to each step
115     real(kind=8) :: hgradient = 0             ! For diffusivity
            ↪ calculations
116     real(kind=8), dimension(nx,ny) :: D = 0  ! For diffusivity
            ↪ calculations
117     real(kind=8), dimension(nx,ny) :: FN = 0 ! For ice flux
            ↪ calculations, on the north side of the point.
118     real(kind=8), dimension(nx,ny) :: FE = 0 ! For ice flux
            ↪ calculations, on the east side of the point.
119     real(kind=8), dimension(nx,ny) :: FS = 0 ! For ice flux
            ↪ calculations, on the south side of the point.
120     real(kind=8), dimension(nx,ny) :: FW = 0 ! For ice flux
            ↪ calculations, on the west side of the point.
121     ! Surface Mass Balance
122     real(kind=8), dimension(nx,ny) :: surface_mass_balance = 0
123     real(kind=8), dimension(nx,ny) :: accumulation = 0
124     real(kind=8), dimension(nx,ny) :: ablation = 0
125     ! Discharge
126     ! Discharge (mass flow) in x direction at each point in m yr^-1
127     real(kind=8), dimension(nx, ny) :: discharge_x = 0
128     ! Discharge (mass flow) in y direction at each point in m yr^-1
129     real(kind=8), dimension(nx, ny) :: discharge_y = 0
130     ! Ice frozen on land (needed for sea level)
131     real(kind=8) :: ice_volume = 0
132
133     ! For later reanalysis: Medium ice elevation at 1m^2
134     real(kind=8), dimension(timesteps) :: H_ts = 0          ! medium
            ↪ ice height at 1m^2 over time
135
136     character(128) :: heartbeat = ''
137     character(256) :: model_description = ''
138
139     ! To get the execution time:
            ↪ http://gcc.gnu.org/onlinedocs/gcc-4.0.4/gfortran/ETIME.html
140     real, dimension(2) :: execution_time ! user, system
141     real :: runtime
142     integer :: last_netcdf_year = 0 ! Last year the netCDF was
            ↪ written. This is used, if the timestep is below one year,
            ↪ that the netCDF is not written several times in this year.
143
144     !=========================
145     ! Input from files
146     !=========================
147
148     ! Landmask/Assignment mask
149     !=========================
150     character(256), parameter :: netcdf_input_bedrock = &
```

```
151         ! 20km resolution
152         !'/home/neff/MasterThesis/data/greenland/topographie/etopo_grl20_landmask
153         !'/home/neff/MasterThesis/data/northhemisphere/topographie/etopo_nhem20_l
154         !'/home/neff/MasterThesis/data/northhemisphere/topographie/ccsm_nhem20_LGI
155         ! 40km resolution
156         '/home/neff/MasterThesis/data/northhemisphere/topographie/etopo_nhem40_lar
157
158     ! The NetCDF Variable name of the bedrock
159     character(256), parameter :: netcdf_input_bedrock_variable =
            ↪ 'LANDMASK' ! 'LANDMASK'
160     ! terrain_mask: 0 = ice, 1 = water, 2 = no ice, 3 = unstable
            ↪ integration
161     integer, dimension(nx, ny) :: terrain_mask = 0    ! "Normal" case
            ↪ (ice) = 0, Water = 1, no ice in this time step = 2
162
163     ! Climate
164     !=========================
165     ! Climate data (temperature, precipitation and the elevation
            ↪ according to the initial temperature)
166     character(128), parameter :: netcdf_input_climate = &
167         ! 20km resolution
168         !'/local_scratch/neff/klima/nh/pd/climate_pd_nh20.nc'
169         !'/local_scratch/neff/klima/nh/pi/climate_pi_nh20.nc'
170         !'/local_scratch/neff/klima/nh/lgm_present_topo/climate_lgm_present_topo_
171         !'/local_scratch/neff/klima/nh/lgm/climate_lgm_nh20.nc'
172         ! 40km resolution
173         !'/data10/neff/ensemble/input/climate/lgm/climate_lgm_nh40.nc'
174         '/data10/neff/ensemble/input/climate/lgm/climate_lgm_tempbias_nh40.nc'
175
176     ! The elevation which is used in the model of the input climate.
            ↪ This is used to calculate the temperature at the sea level.
177     ! For the ReAnalyse data, the elevation of the ice surface is
            ↪ used. Please also adjust the
            ↪ 'netcdf_input_climate_elevation_variable' variable.
178     character(256), parameter :: netcdf_input_climate_elevation = &
179         ! 20km resolution
180         !'/home/neff/MasterThesis/data/northhemisphere/topographie/etopo_nhem20_l
181         !'/home/neff/MasterThesis/data/northhemisphere/topographie/ccsm_nhem20_pr
182         !'/home/neff/MasterThesis/data/northhemisphere/topographie/ccsm_nhem20_LGI
183         ! 40km resolution
184         '/home/neff/MasterThesis/data/northhemisphere/topographie/ccsm_nhem40_LGM
185
186     ! Variable name in the netcdf_input_climate_elevation file which
            ↪ is used for the elevation.
187     character(256), parameter ::
            ↪ netcdf_input_climate_elevation_variable = 'SURFACE'
            ↪ !'ICE_NHEM20' ! ICE_NHEM20 = Northhemisphere for PD,
            ↪ ICE_GRL20 = Greenland
188     real(kind=8), dimension(nx, ny, 365) :: temperature = 0
            ↪              ! Temperature in Kelvin at the ice elevation
            ↪ for 365 days to calculate the accumulation and ablation out
            ↪ of it.
189     real(kind=8), dimension(nx, ny, 365) :: potential_temperature = 0
            ↪    ! Temperature in Kelvin at the sea level for 365 days to
            ↪ calculate the accumulation and ablation out of it.
190     real(kind=8), dimension(nx,ny) :: initial_ice_elevation = 0
            ↪          ! Initial elevation of the ice, where the
```

```
        ↪ temperature is measured
191     real(kind=8), dimension(nx, ny, 365) :: precipitation = 0
        ↪           ! Precipitation (m/yr) for 365 days to calculate
        ↪ the accumulation and ablation out of it.
192
193     ! Initial state
194     !=========================
195     ! Read initial state (this can be controlled with the flag
        ↪ read_initial_state)
196     character(128), parameter :: initial_netcdf_file =
        ↪ '/local_scratch/neff/model/modelinput/input.nc'  ! The
        ↪ initial netcdf file with the the height and bedrock, no
        ↪ time axes!
197     character(128), parameter :: initial_bedrock_variable_name =
        ↪ 'bedrock'              ! The name of the bedrock variable
198     character(128), parameter :: initial_height_variable_name =
        ↪ 'height'                ! The name of the height variable
199
200     !=========================
201     ! Output to files
202     !=========================
203     ! Output variables, with "realistic" values, values of the
        ↪ bedrock is not 0
204     real(kind=8), dimension(nx, ny) :: bedrock_netcdf      ! The
        ↪ bedrock, with values below 0 on the water
205     real(kind=8), dimension(nx, ny) :: elevation_netcdf    !
        ↪ Elevation with ice: variable 'elvevation' on land,
        ↪ bedrock_netcdf in the water
206
207     ! netCDF
208     !-------
209     character(512) :: output_directory =
        ↪ '/local_scratch/neff/model/modeloutput/'
210     character(128), parameter :: netcdf_output_filename = 'ice.nc'
211     integer :: filehandle_netcdf                    ! cant be a
        ↪ parameter, cause the netcdf functions modifies it.
212     integer, parameter :: netcdf_timesteps = 500    ! Log every
        ↪ <value> years the values to the netCDF
213     ! Variable IDs: height, bed
214     integer :: height_varid       ! Variable for the netCDF Height,
        ↪ parameter
215     integer :: bedrock_varid      ! Variable for the netCDF Bedrock,
        ↪ parameter
216     integer :: acc_varid          ! Variable for the netCDF
        ↪ accumulation, parameter
217     integer :: abl_varid          ! Variable for the netCDF ablation,
        ↪ parameter
218     integer :: diffusivity_varid  ! Variable for the netCDF
        ↪ diffusivity per year, parameter
219     integer :: discharge_x_varid  ! Variable for the netCDF discharge
        ↪ in x direction per year, parameter
220     integer :: discharge_y_varid  ! Variable for the netCDF discharge
        ↪ in y direction per year, parameter
221     integer :: terrain_mask_varid ! Variable for the netCDF
        ↪ terrain_mask, parameter
222
223 end module variables
```

IceModel.f90

The file `IceModel.f90` is the main application.

```
! IceBern2D model in Fortran.
! Two-dimensional flow line ice sheet model using Glen's law
!!
! Some (not all!) references:
!-------------------------------------------------------
! Oerlemans, J. (1981), Some basic experiments with a vertically-integrated ice sheet model, Tellus 33, 1-11
! Oerlemans, J. (1982), Glacial Cycles and Ice-Sheet Modelling, Climatic Change 4, 353-374
! Huybrechts et al. (1996), The EISMINT benchmarks for testing ice-sheet models, Annals of Glaciology, 23, 1-12
!
! Author:        neff@climate.unibe.ch
! Date:          2014/05
! Revision:      1.0b
! SVN Info:      $Id: IceModel.f90 189 2014-06-01 15:21:53Z basil $
!
! I acknowledge everyone who acknowledges me using this source code as base of a future model.

program IceModel
    !=========================
    ! Include Modules
    !=========================
    use variables        ! Module with all variables
    use io_read          ! Own module to read the values
    use io_write         ! Own module to write the values
    use common           ! Common module
    use smb              ! Own module to get the values from the external forcing

    ! output directory
    model_description = TRIM(adjustl(experiment_description))
    if (store_input_netcdf .or. write_netcdf) then
        call init_output_directory(output_directory, model_description)
    endif

    !=========================
    ! initialze Variable
    !=========================
    if(debug > 0 ) then
        print *, "Initialize Variables from external files"
        print *, "----------------------------------------"
    end if

    ! If the sea level gets not adjusted, set the offset to 0
    if (.not.adjust_sea_level) then
        sea_level_offset = 0
    end if

    ! If the bedrock is defined in a NetCDF file,
    if (read_bedrock) then
```

```
48          if(debug > 1 ) then
49              print *, "*** Read '",
                    ↪ TRIM(adjustl(netcdf_input_bedrock_variable)) ,"'
                    ↪ from file: ", TRIM(adjustl(netcdf_input_bedrock))
50          end if
51          ! Bedrock_initial is how the bedrock would be without iceload.
52          Bedrock_initial = read_variable(netcdf_input_bedrock, nx, ny,
                ↪ TRIM(adjustl(netcdf_input_bedrock_variable)))
53
54          if(debug > 0) then
55              print *, "*** Get terrain_mask from the initial bedrock."
56          end if
57          terrain_mask = read_watermask(Bedrock_initial, nx, ny,
                ↪ (sea_level + sea_level_offset), Bedrock_initial,
                ↪ ice_thickness) ! 1 = water, 0 = normal case, 3 =
                ↪ unstable grid points
58
59          if(store_input_netcdf) then
60              if(debug > 0) then
61                  print *, "*** Copy bedrock input (",
                        ↪ TRIM(adjustl(netcdf_input_bedrock)) , ") file
                        ↪ to the output directory: ",
                        ↪ TRIM(adjustl(output_directory))
62              end if
63              call copy_to_output_directory(netcdf_input_bedrock,
                    ↪ output_directory)
64          end if
65
66          ! set the initial value
67          Bedrock = Bedrock_initial
68          bedrock_netcdf = Bedrock_initial
69          elevation_netcdf = Bedrock_initial
70          ! Set the height of the bedrock on the water to the sea
                ↪ level, Otherwise there will be an unstable integration,
71          ! cause the flux at the coast will get very high
72          do ix=1,nx,1
73              do iy=1,ny,1
74                  if (terrain_mask(ix,iy) == 1) then
75                      Bedrock(ix,iy) = sea_level + sea_level_offset
76                      Ice_thickness(ix,iy) = 0
77                  endif
78              end do
79          end do
80      endif
81
82      ! Read the initial state out of an NetCDF File
83      ! For this the watermask from the bedrock is used!
84      if (read_initial_state) then
85          ! Store the input files
86          if(store_input_netcdf) then
87              call copy_to_output_directory(initial_netcdf_file,
                    ↪ output_directory)
88          end if
89
90          ! Bedrock_initial is how the bedrock would be without iceload.
91          ! already done before, so not needed twice
```

```
92          ! Bedrock_initial = read_variable(netcdf_input_bedrock, nx,
               ↪ ny, TRIM(adjustl(netcdf_input_bedrock_variable)))
93
94          if(debug > 0 ) then
95              print *, '*** Read bedrock state from file: ',
                   ↪ TRIM(adjustl(initial_netcdf_file))
96          end if
97          Bedrock = read_variable(initial_netcdf_file, nx, ny,
               ↪ initial_bedrock_variable_name)
98
99          elevation_netcdf = read_variable(initial_netcdf_file, nx, ny,
               ↪ initial_height_variable_name)
100         elevation = elevation_netcdf
101
102         Ice_thickness = elevation - Bedrock
103
104         ! Restore Sea level
105         sea_level = get_sea_level(Ice_thickness, nx, ny, dx, dy,
               ↪ ocean_area)
106
107         ! Assign the terrain_mask
108         ! 1 = water, 0 = normal case, 3 = unstable grid points
109         terrain_mask = read_watermask(Bedrock_initial, nx, ny,
               ↪ (sea_level + sea_level_offset), bedrock_initial,
               ↪ ice_thickness)
110
111         ! Set the height of the bedrock on the water to the sea
               ↪ level, Otherwise there will be an unstable integration,
112         ! cause the flux at the coast will get very high
113         do ix=1,nx,1
114             do iy=1,ny,1
115                 if (terrain_mask(ix,iy) == 1) then
116                     Bedrock(ix,iy) = sea_level + sea_level_offset
117                     Ice_thickness(ix,iy) = 0
118                 endif
119             end do
120         end do
121     end if  ! ENDIF: Read initial state
122
123     ! Elevation of ice surface above sea level (over the water:
           ↪ Bedrock and Ice_thickness are 0 -> elevation = 0)
124     elevation = Bedrock + Ice_thickness
125     ! Distance of each grid box from bottom domain boundary
126     y = y * dy
127     ! Distance of each grid box from left domain boundary
128     X = x * dx
129
130     ! Accumulation
131     temperature = read_climate(netcdf_input_climate, nx, ny,
           ↪ 'TEMPERATURE')
132     initial_ice_elevation =
           ↪ read_variable(netcdf_input_climate_elevation, nx, ny,
           ↪ TRIM(adjustl(netcdf_input_climate_elevation_variable)))
133     ! change the temperature for every day to potential temperature.
           ↪ Substract the temperature diffence to the sea level.
134     do id=1,365,1
```

```
135         potential_temperature(:,:,id) = temperature(:,:,id) +
               ↪ (initial_ice_elevation * lapse_rate)
136         ! Adjust temperature to elevation!
137         temperature(:,:,id) = potential_temperature(:,:,id) -
               ↪ (elevation * lapse_rate)
138     end do
139     ! precipitation
140     precipitation = read_climate(netcdf_input_climate, nx, ny,
           ↪ 'PRECIPITATION')
141     accumulation = temperature_dependent_accumulation(nx, ny,
           ↪ temperature, precipitation, precipitation_unit,
           ↪ accumulation_daily_temperature_threshold)
142     if(ignore_himalaya) then
143         call ignore_himalaya_accumulation(accumulation, nx, ny)
144     end if
145     ablation = temperature_dependent_ablation(nx, ny, temperature,
           ↪ beta)
146     surface_mass_balance = accumulation - ablation
147     ! store the data
148     if(store_input_netcdf) then
149         call copy_to_output_directory(netcdf_input_climate,
               ↪ output_directory)
150     end if
151
152     ! NETCDF
153     if (write_netcdf) then
154         call init_netcdf_file(TRIM(adjustl(output_directory)) //
               ↪ netcdf_output_filename, filehandle_netcdf, ny, nx, dy,
               ↪ dx, height_varid, bedrock_varid, acc_varid, abl_varid,
               ↪ diffusivity_varid, discharge_x_varid,
               ↪ discharge_y_varid, terrain_mask_varid)
155
156         ! Write initial values as year 0 to the netcdf
157         ! but only height and bedrock, since the other values do not
               ↪ exist
158         call writeNCDFGridValues(filehandle_netcdf, 1, bedrock_varid,
               ↪ bedrock_netcdf(1:nx,1:ny), ny, nx)
159
160         call writeNCDFGridValues(filehandle_netcdf, 1,
               ↪ height_varid,elevation_netcdf(1:nx,1:ny), ny, nx)
161
162         call writeNCDFGridIntegerValues(filehandle_netcdf,
               ↪ (myyear(it)/netcdf_timesteps) + 1,
               ↪ terrain_mask_varid,terrain_mask(1:nx,1:ny), ny, nx)
163
164         ! Accumulation and Ablation
165         call writeNCDFGridValues(filehandle_netcdf,
               ↪ (myyear(it)/netcdf_timesteps) + 1, acc_varid,
               ↪ accumulation(1:nx, 1:ny) * seconds_per_year, ny, nx)
166         call writeNCDFGridValues(filehandle_netcdf,
               ↪ (myyear(it)/netcdf_timesteps) + 1, abl_varid,
               ↪ ablation(1:nx, 1:ny) * seconds_per_year, ny, nx)
167     endif
168
169
170     !=========================
171     ! Lets go, do the loop
```

```
172     !=========================
173     if(debug > 0) then
174         print *, "Loop over time steps"
175         print *, "--------------------"
176     endif
177     do
178
179         ! Check if the loop conditions are at the end
180         if (myyear(it) >= maxyears) then
181             print *, 'The end is near: ', myyear(it)
182             exit ! Jumps out of the loop. Does not exit the
                    ↪ application (call exit(1)), otherwise the script is
                    ↪ not terminated correctly
183         end if
184
185         ! Time series of diagnostics
186         myyear(it) = int(it * real(dt)/(3600*24*365))
187
188         ! Print Heartbeat
189         if ((mod(myyear(it), netcdf_timesteps) == 0) .and. (debug >
                ↪ 0) .and. (last_netcdf_year .lt. myyear(it)) ) then
190             call ETIME(execution_time, runtime)
191             Write( heartbeat, '(i8)' ) myyear(it)
192             print *, 'Year: ', TRIM(adjustl(heartbeat)), ', Runtime
                    ↪ [s]: ', int(runtime), ', End in [s]: ', int((
                    ↪ (runtime/myyear(it)) * maxyears) - runtime), ',
                    ↪ yr/hour: ', (myyear(it)/runtime * 60*60), ', Sea
                    ↪ level: ', (sea_level + sea_level_offset), ', NetCDF
                    ↪ TS: ', ((myyear(it)/netcdf_timesteps) + 1)
193         end if
194
195         ! Adjust Sea Level every 50 years
196         if(adjust_sea_level) then
197             if (mod(myyear(it), 50) == 0) then
198                 sea_level = get_sea_level(ice_thickness, nx, ny, dx,
                        ↪ dy, ocean_area)
199                 ! Possible improvement: Do not take the initial
                        ↪ bedrock, use the current one but forget the ice
                        ↪ above it (and it should still increase, even if
                        ↪ it is below the water
200                 ! But this could get to complicated, cause we do not
                        ↪ want any sea in the middle of america.
201                 ! (use bedrock_netcdf, cause the bedrock is not equal
                        ↪ to the sea level)
202                 terrain_mask = read_watermask(Bedrock_initial, nx,
                        ↪ ny, (sea_level + sea_level_offset),
                        ↪ Bedrock_netcdf, ice_thickness)
203                 ! Set the sea level in the netcdf file for every
                        ↪ water point
204                 where(terrain_mask(:,:) .eq. 1)
205                     Bedrock(:,:) = sea_level + sea_level_offset
206                     Ice_thickness(:,:) = 0
207                 end where
208             end if
209         end if
210
211         ! compute diffusivity
```

```
212         ! first loop over x-axes
213         do ix=1,nx,1
214             ! then loop over y-axes
215             do iy=1,ny,1
216                 ! South border
217                 if (iy == 1) then
218                     ! Special Case, the corner in the east and west
219                     ! north west corner
220                     if (ix == 1) then
221                         hgradient = abs( ( ((elevation(ix,iy) -
                            ↪ elevation(ix+1,iy)) / real(dx,8))**2 )
                            ↪ + ( ((elevation(ix,iy+1) -
                            ↪ elevation(ix,iy))/real(dy,8))**2 ) )
222                     ! north east corner
223                     else if(ix == nx) then
224                         hgradient = abs( ( ((elevation(ix-1,iy) -
                            ↪ elevation(ix,iy)) / real(dx,8))**2 ) +
                            ↪ ( ((elevation(ix,iy+1) -
                            ↪ elevation(ix,iy))/real(dy,8))**2 ) )
225                     ! South border
226                     else
227                         hgradient = abs( ( ((elevation(ix-1,iy) -
                            ↪ elevation(ix+1,iy)) / (2 * dx))**2 ) +
                            ↪ ( ((elevation(ix,iy+1) -
                            ↪ elevation(ix,iy))/dy)**2 ) )
228                     endif
229                 ! North border
230                 else if (iy == ny) then
231                     ! north west corner
232                     if (ix == 1) then
233                         hgradient = abs( ( ((elevation(ix,iy) -
                            ↪ elevation(ix+1,iy)) / dx)**2 ) + (
                            ↪ ((elevation(ix,iy) -
                            ↪ elevation(ix,iy-1))/dy)**2 ) )
234                     ! north east corner
235                     else if(ix == nx) then
236                         hgradient = abs( ( ((elevation(ix-1,iy) -
                            ↪ elevation(ix,iy)) / dx)**2 ) + (
                            ↪ ((elevation(ix,iy) -
                            ↪ elevation(ix,iy-1))/dy)**2 ) )
237                     ! North Border
238                     else
239                         hgradient = abs( ( ((elevation(ix-1,iy) -
                            ↪ elevation(ix+1,iy)) / (2*dx))**2 ) + (
                            ↪ ((elevation(ix,iy) -
                            ↪ elevation(ix,iy-1))/dy)**2 ) )
240                     endif
241                 ! East Border
242                 else if (ix == 1) then
243                     hgradient = abs( ( ((elevation(ix,iy) -
                        ↪ elevation(ix+1,iy)) / dx)**2 ) + (
                        ↪ ((elevation(ix,iy+1) -
                        ↪ elevation(ix,iy-1))/(2*dy))**2 ) )
244                 ! West Border
245                 else if (ix == nx) then
246                     ! wrong
```

```
247                     hgradient = abs( ( ((elevation(ix-1,iy) -
                            ↪ elevation(ix,iy)) / dx)**2 ) + (
                            ↪ ((elevation(ix,iy+1) -
                            ↪ elevation(ix,iy-1))/(2*dy))**2 ) )
248                 ! the normal case in the field
249                 else
250                     ! Discretization with full steps (x-1 and x+1)
251                     ! Do the check with the terrain_mask only in the
                            ↪ normal case, otherwise it will get to
                            ↪ confising in the code. And most of the
                            ↪ cells are anyway not at the border.
252                     ! Calculate the gradient only, if there is at
                            ↪ least one grid point with a zero in the 3x3
                            ↪ grid around the center.
253                     if(minval(terrain_mask((ix-1):(ix+1),(iy-1):(iy+1)))
                            ↪ == 0) then
254                         hgradient = abs( ( ((elevation(ix-1,iy) -
                                ↪ elevation(ix+1,iy)) / (2 * dx))**2 ) +
                                ↪ ( ((elevation(ix, iy+1) - elevation(ix,
                                ↪ iy-1))/(2 * dy))**2 ) )
255
256                         ! If the hgradient is zero or very close to
                                ↪ it (one point island, cause the ocean
                                ↪ has elevation 0), the ice gets
                                ↪ accumulated to infinity until the model
                                ↪ gives an unstable integration.
257                         ! In this case, the gradient is calculated
                                ↪ with the elevation of the sea floor.
258                         ! This may not be very realistic (since the
                                ↪ sea floor does not interact with the
                                ↪ ice), but it only affects small islands.
259                         if (abs(hgradient) .lt. 1E-014) then
260                             hgradient = abs( (
                                    ↪ ((elevation_netcdf(ix-1,iy) -
                                    ↪ elevation_netcdf(ix+1, iy)) / (2 *
                                    ↪ dx))**2 ) + (
                                    ↪ ((elevation_netcdf(ix, iy+1) -
                                    ↪ elevation_netcdf(ix, iy-1))/(2 *
                                    ↪ dy))**2 ) )
261                         end if
262                     else
263                         hgradient = 0
264                         ! DEBUG
265                         if(debug > 6 ) then
266                             print *,it,' - hgradient not calculated
                                    ↪ at grid point: ', ix, iy, ',
                                    ↪ terrain_mask: ', terrain_mask(ix,iy)
267                         endif
268                     end if
269                 end if
270
271                 ! Diffusivity calculation at point ix, iy
272                 ! Eismint (Huybrechts, 1996): equation 3 (first part,
                        ↪ without H gradient: This part is multiplied
                        ↪ later)
273                 if(hgradient .ne. 0) then ! Check if hgradient is
                        ↪ calculated before
```

```
274                     D(ix,iy) = ((2 * (ice_flux_adjustments *
                            ↪ A_Eismint) * (rho_ice * g) **
                            ↪ n_Eismint)/(n_Eismint + 2)) *
                            ↪ (Ice_thickness(ix, iy)**(n_Eismint + 2)) *
                            ↪ (hgradient**((n_Eismint - 1)/2))
275                 else
276                     D(ix, iy) = 0
277                 endif
278             end do ! loop over y-axes
279         end do ! loop over x-axes
280
281         ! Write NetCDF with Diffusivity
282         if (write_netcdf) then
283             ! only write in specific timesteps (netcdf_timesteps) to
                    ↪ the netCDF File
284             if (mod(myyear(it), netcdf_timesteps) == 0 .and.
                    ↪ (last_netcdf_year .lt. myyear(it))) then
285                 call writeNCDFGridValues(filehandle_netcdf,
                        ↪ (myyear(it)/netcdf_timesteps) + 1,
                        ↪ diffusivity_varid, D(1:nx, 1:ny) *
                        ↪ seconds_per_year, ny, nx)
286             end if
287         end if
288
289         ! Compute Ice flux
290         ! first loop over x-axes
291         do ix=1,nx,1
292             ! then loop over y-axes
293             do iy=1,ny,1
294                 ! Default, all are 0, which is the value at the
                        ↪ border and corners.
295                 ! Set them later, if the position is away from the
                        ↪ border
296                 FS(ix,iy) = 0
297                 FE(ix,iy) = 0
298                 FN(ix,iy) = 0
299                 FW(ix,iy) = 0
300
301                 ! North/South Direction
302                 if (iy > 1 .AND. iy < ny) then
303                     ! Check if there is at least one point with ice
304                     if(minval(terrain_mask(ix,iy:(iy+1))) == 0) then
305                         FN(ix,iy) = 0.5 * ((D(ix,iy + 1) + D(ix,iy))
                                ↪ * ((elevation(ix,iy + 1) -
                                ↪ elevation(ix,iy))/real(dy,8) ))
306                         ! Check if Flux is higher than the available
                                ↪ amount of ice, reset it to the amount
                                ↪ of ice.
307                         if(FN(ix,iy) .gt. 0) then ! Positiv = Into
                                ↪ the grid point
308                             FN(ix,iy) = min((Ice_thickness(ix,iy +
                                    ↪ 1)*dy/dt),FN(ix,iy))
309                         else ! Negativ = Away from the gridpoint
310                             FN(ix,iy) =
                                    ↪ max(-(Ice_thickness(ix,iy)*dy/dt),FN(ix,iy))
311                         end if
312                     endif
```

```
313                     ! Check if there is at least one point with ice
314                     if(minval(terrain_mask(ix,(iy-1):iy)) == 0) then
315                         FS(ix,iy) = 0.5 * real((D(ix,iy) + D(ix,iy -
                                ↪ 1)) * ((elevation(ix,iy) -
                                ↪ elevation(ix,iy - 1))/real(dy) ))
316                         ! Check if Flux is higher than the available
                                ↪ amount of ice, reset it to the amount
                                ↪ of ice.
317                         if(FS(ix,iy) .gt. 0) then ! Positiv = Away
                                ↪ from the grid point
318                             FS(ix,iy) =
                                    ↪ min((Ice_thickness(ix,iy)*dy/dt),FS(ix,iy))
319                         else ! Negativ = Into the grid point
320                             FS(ix,iy) =
                                    ↪ max(-(Ice_thickness(ix,iy-1)*dy/dt),FS(ix,iy))
321                         end if
322                     endif
323                 end if
324                 ! East/West Direction
325                 if (ix > 1 .AND. ix < nx) then
326                     ! Check if there is at least one point with ice
327                     if(minval(terrain_mask(ix:(ix + 1),iy)) == 0) then
328                         FE(ix,iy) = 0.5 * ((D(ix + 1,iy) + D(ix,iy))
                                ↪ * ((elevation(ix + 1,iy) -
                                ↪ elevation(ix,iy))/real(dx,8) ))
329                         ! Check if Flux is higher than the available
                                ↪ amount of ice, reset it to the amount
                                ↪ of ice.
330                         if(FE(ix,iy) .gt. 0) then ! Positiv = Into
                                ↪ the grid point
331                             FE(ix,iy) = min((Ice_thickness(ix +
                                    ↪ 1,iy)*dx/dt),FE(ix,iy))
332                         else ! Negativ = Away from the grid point
333                             FE(ix,iy) =
                                    ↪ max(-(Ice_thickness(ix,iy)*dx/dt),FE(ix,iy))
334                         end if
335                     endif
336                     ! Check if there is at least one point with ice
337                     if(minval(terrain_mask((ix-1):ix,iy)) == 0) then
338                         FW(ix,iy) = 0.5 * real((D(ix,iy) + D(ix -
                                ↪ 1,iy)) * ((elevation(ix,iy) -
                                ↪ elevation(ix - 1,iy))/real(dx) ))
339                         ! Check if Flux is higher than the available
                                ↪ amount of ice, reset it to the amount
                                ↪ of ice.
340                         if(FW(ix,iy) .gt. 0) then ! Positiv = Away
                                ↪ from the grid point
341                             FW(ix,iy) =
                                    ↪ min((Ice_thickness(ix,iy)*dx/dt),FW(ix,iy))
342                         else ! Negativ = Into the gridpoint
343                             FW(ix,iy) = max(-(Ice_thickness(ix -
                                    ↪ 1,iy)*dx/dt),FW(ix,iy))
344                         end if
345                     endif
346                 end if
347             end do ! loop over y-axes
348         end do ! loop over x-axes
```

```
349
350         ! compute mass flow (discharge)
351         ! This is only done for analytical reasons (during the
                ↪ eismint test), to check the values in the netcdf file.
352         ! The values are not needed for later calculatieons.
353         if (write_netcdf) then
354             ! only write in specific timesteps (netcdf_timesteps) to
                    ↪ the netCDF File
355             if (mod(myyear(it), netcdf_timesteps) == 0 .and.
                    ↪ (last_netcdf_year .lt. myyear(it))) then
356                 discharge_x(:, :) = FE(:,:) * seconds_per_year
                        ↪ !(((real(FE(ix,iy)) - FW(ix,iy))/real(dx)) *
                        ↪ seconds_per_year)
357                 discharge_y(:, :) = FN(:,:) * seconds_per_year !
                        ↪ (((real(FN(ix,iy)) - FS(ix,iy))/real(dy)) *
                        ↪ seconds_per_year)
358                 ! Write it to the NetCDF file
359                 call writeNCDFGridValues(filehandle_netcdf,
                        ↪ (myyear(it)/netcdf_timesteps) + 1,
                        ↪ discharge_x_varid, discharge_x(1:nx, 1:ny), ny,
                        ↪ nx)
360                 call writeNCDFGridValues(filehandle_netcdf,
                        ↪ (myyear(it)/netcdf_timesteps) + 1,
                        ↪ discharge_y_varid, discharge_y(1:nx, 1:ny) ,
                        ↪ ny, nx)
361             end if
362         end if
363
364         ! Accumulation
365         ! Calculate the elevation feedback only every 10 years.
366         if (mod(myyear(it), 10) == 0) then
367             ! add the lapse rate for every day
368             do id = 1,365,1
369                 temperature(:,:,id) = potential_temperature(:,:,id) -
                        ↪ (elevation * lapse_rate)
370             end do
371             accumulation = temperature_dependent_accumulation(nx, ny,
                    ↪ temperature, precipitation, precipitation_unit,
                    ↪ accumulation_daily_temperature_threshold)
372             ablation = temperature_dependent_ablation(nx, ny,
                    ↪ temperature, beta)
373             if(ignore_himalaya) then
374                 call ignore_himalaya_accumulation(accumulation, nx,
                        ↪ ny)
375             end if
376             surface_mass_balance = accumulation - ablation
377         endif
378
379         ! Write NetCDF with surface mass balance
380         if (write_netcdf) then
381             ! only write in specific timesteps (netcdf_timesteps) to
                    ↪ the netCDF File
382             if (mod(myyear(it), netcdf_timesteps) == 0 .and.
                    ↪ (last_netcdf_year .lt. myyear(it))) then
383                 call writeNCDFGridValues(filehandle_netcdf,
                        ↪ (myyear(it)/netcdf_timesteps) + 1, acc_varid,
                        ↪ accumulation(1:nx, 1:ny) * seconds_per_year,
```

```
                    ↪ ny, nx)
384                 call writeNCDFGridValues(filehandle_netcdf,
                        ↪ (myyear(it)/netcdf_timesteps) + 1, abl_varid,
                        ↪ ablation(1:nx, 1:ny) * seconds_per_year, ny, nx)
385             end if
386         end if
387
388
389         ! Compute new ice thickness
390         ! first loop over x-axes
391         do ix=1,nx,1
392             ! then loop over y-axes
393             do iy=1,ny,1
394                 ! Height
395                 ! Only do it where there is no water
396                 if (terrain_mask(ix, iy) .ne. 1) then
397                     Ice_thickness(ix,iy) = max(real(0,8),
                            ↪ (Ice_thickness(ix,iy) &
398                         ! Flux in North/South direction
399                         + (((FN(ix,iy) - FS(ix,iy))/dy) * dt) &
400                         ! Flux in East/West direction
401                         + (((FE(ix,iy) - FW(ix,iy))/dx) * dt) &
402                         ! Accumulation
403                         + (surface_mass_balance(ix,iy) * dt)) )
404                     !
405                     ! If the new ice_thickness is 0, set the
                            ↪ terrain_mask to 2
406                     if(Ice_thickness(ix,iy) == 0) then
407                         ! DEBUG
408                         if(debug > 6 ) then
409                             print *,it,' - Assigned value 2 at grid
                                    ↪ point: ', ix, iy, ', height: ',
                                    ↪ Ice_thickness(ix,iy)
410                         endif
411                         terrain_mask(ix,iy) = 2
412                     else
413                         terrain_mask(ix,iy) = 0
414                     endif
415                 endif
416             end do ! loop over y-axes
417         end do ! loop over x-axes
418
419
420         ! Compute bedrock sinking
421         if (active_bedrock) then
422             ! first loop over x-axes
423             do ix=1,nx,1
424                 ! then loop over y-axes
425                 do iy=1,ny,1
426                     Bedrock(ix,iy) = bedrock_netcdf(ix,iy) -
                            ↪ ((real(1,8)/3 * Ice_thickness(ix,iy)) +
                            ↪ (bedrock_netcdf(ix,iy) -
                            ↪ Bedrock_Initial(ix,iy)) ) *
                            ↪ (real(dt,8)/tstar)
427                     ! bedrock_netcdf = Real bedrock, with negative
                            ↪ values in the ocean (Bedrock is set to the
                            ↪ sealevel on the ocean).
```

```
428                     ! The variable "Bedrock" is only used for
                           ↪ calculation of hgradient. Otherwise
                           ↪ bedrock_netcdf is used.
429                     ! For understanding reasons, it would be usefull
                           ↪ to set the "Bedrock" at a water grid cell
                           ↪ right here to sealevel.
430                     bedrock_netcdf(ix, iy) = Bedrock(ix, iy)
431                     if(read_bedrock) then
432                         ! Elevation for netcdf output, but with
                               ↪ negative values on the water
433                         elevation_netcdf(ix, iy) = Bedrock(ix, iy) +
                               ↪ Ice_thickness(ix, iy)
434                     endif
435                     ! if the bedrock is read from a netcdf file, set
                           ↪ the ice height on the water to zero
436                     if((terrain_mask(ix, iy) == 1) .and.
                           ↪ (read_bedrock)) then
437                      Ice_thickness(ix,iy) = 0
438                     endif
439                 end do
440             end do
441         endif
442
443         ! Write also the bedrock to the NetCDF file, if bedrock
               ↪ sinking is not enabled
444         if (write_netcdf) then
445             ! only write in specific timesteps to the netCDF File
446             if (mod(myyear(it), netcdf_timesteps) == 0 .and.
                   ↪ (last_netcdf_year .lt. myyear(it))) then
447              call writeNCDFGridValues(filehandle_netcdf,
                       ↪ (myyear(it)/netcdf_timesteps) + 1,
                       ↪ bedrock_varid, bedrock_netcdf(1:nx,1:ny), ny,
                       ↪ nx)
448             end if
449         end if
450
451         ! If it is a domain, without any special bedrock, reset the
               ↪ ice thickness at the border
452         if (.not.read_bedrock) then
453             ! set ice thickness at the edge to zero
454             ! North and South Border
455             Ice_thickness(1:nx,1) = 0
456             Ice_thickness(1:nx,ny) = 0
457             ! East and West Border
458             Ice_thickness(1,1:ny) = 0
459             Ice_thickness(nx,1:ny) = 0
460             ! Elevation (similar to variable z) for output, but with
                   ↪ negative values over water
461             elevation_netcdf = Bedrock + Ice_thickness
462         endif
463
464         ! update prognostic variable
465         ! This is done for the output some lines before
466         elevation = Bedrock + Ice_thickness
467
468
469         ! Time series of diagnostics
```

```
470         ! (max(1,size(Ice_thickness)))) = Amount of Grid Boxes
471         ! real((real(L,8) * dx * W * dy),8) ) = square meters
472         H_ts(it) = ( (sum(Ice_thickness))/real((real(L,8) * dx * W *
                ↪ dy),8) ) ! Average height at 1m^2: mean(H)
473
474         if (write_netcdf) then
475             ! only write in specific timesteps to the netCDF File
476             if (mod(myyear(it), netcdf_timesteps) == 0 .and.
                    ↪ (last_netcdf_year .lt. myyear(it))) then
477                 ! reverse the order, so that y = 0 is in the north:
                        ↪ B(1:nx,ny:1:-1)
478                 ! Change order of arrays:
                        ↪ http://kiwi.atmos.colostate.edu/fortran/docs/fortran2012.key
479                 call writeNCDFGridValues(filehandle_netcdf,
                        ↪ (myyear(it)/netcdf_timesteps) + 1,
                        ↪ height_varid,elevation_netcdf(1:nx,1:ny), ny,
                        ↪ nx)
480                 ! Write terrain mask to netCDF
481                 call writeNCDFGridIntegerValues(filehandle_netcdf,
                        ↪ (myyear(it)/netcdf_timesteps) + 1,
                        ↪ terrain_mask_varid,terrain_mask(1:nx,1:ny), ny,
                        ↪ nx)
482                 ! this is the last "official" netCDF access in this
                        ↪ loop. Assign the year to the last_netcdf_year
                        ↪ variable.
483                 last_netcdf_year = myyear(it)
484             end if
485         end if
486
487
488         ! Integration check: H-ts(it) = mean ice thickness at 1m^2
489         if (H_ts(it) > 10000 ) then ! 10000m of ice on every 1m^2
490             print *, 'unstable integration!'
491             print *, '----------------------'
492             print *, 'Year: ', myyear(it), it
493             print *, 'H_ts(it): ', H_ts(it)
494             print *, 'Sea Level: ', sea_level + sea_level_offset
495
496             ! Get unstable points and assign it in the terrain_mask
                    ↪ to value 3
497             ! x-axes
498             do ix=1,nx,1
499                 ! then loop over y-axes
500                 do iy=1,ny,1
501                     if (Ice_thickness(ix,iy) > 6000) then
502                         terrain_mask(ix:ix,iy:iy) = 3
503                     endif
504                 end do ! y axes
505             end do ! x axes
506
507             ! Write last values to the NetCDF file and and close it.
508             if (write_netcdf) then
509                 ! Calculate/Refresh discharge
510                 discharge_x(:, :) = FE(:,:) * seconds_per_year
511                 discharge_y(:, :) = FN(:,:) * seconds_per_year
512
513                 ! Write all variables for later reanalyse
```

```
514                 call writeNCDFGridValues(filehandle_netcdf,
                        ↪ (last_netcdf_year/netcdf_timesteps) +1,
                        ↪ bedrock_varid, bedrock_netcdf(1:nx,1:ny), ny,
                        ↪ nx)
515                 call writeNCDFGridValues(filehandle_netcdf,
                        ↪ (last_netcdf_year/netcdf_timesteps) + 1,
                        ↪ height_varid,elevation_netcdf(1:nx,1:ny), ny,
                        ↪ nx)
516                 call writeNCDFGridIntegerValues(filehandle_netcdf,
                        ↪ (last_netcdf_year/netcdf_timesteps) + 1,
                        ↪ terrain_mask_varid,terrain_mask(1:nx,1:ny), ny,
                        ↪ nx)
517                 call writeNCDFGridValues(filehandle_netcdf,
                        ↪ (last_netcdf_year/netcdf_timesteps) + 1,
                        ↪ diffusivity_varid, D(1:nx, 1:ny) *
                        ↪ seconds_per_year, ny, nx)
518                 call writeNCDFGridValues(filehandle_netcdf,
                        ↪ (last_netcdf_year/netcdf_timesteps) + 1,
                        ↪ discharge_x_varid, discharge_x(1:nx, 1:ny) ,
                        ↪ ny, nx)
519                 call writeNCDFGridValues(filehandle_netcdf,
                        ↪ (last_netcdf_year/netcdf_timesteps) + 1,
                        ↪ discharge_y_varid, discharge_y(1:nx, 1:ny) ,
                        ↪ ny, nx)
520                 call writeNCDFGridValues(filehandle_netcdf,
                        ↪ (myyear(it)/netcdf_timesteps) + 1, acc_varid,
                        ↪ accumulation(1:nx, 1:ny) * seconds_per_year,
                        ↪ ny, nx)
521                 call writeNCDFGridValues(filehandle_netcdf,
                        ↪ (myyear(it)/netcdf_timesteps) + 1, abl_varid,
                        ↪ ablation(1:nx, 1:ny) * seconds_per_year, ny, nx)
522                 call closeNCDFFile(filehandle_netcdf)
523             endif
524             ! Runtime information
525             call ETIME(execution_time, runtime)
526             print *, "Runtime [mm:ss]: ", int(runtime/60), ':',
                    ↪ mod(int(runtime),60)
527             print *, 'Year: ', myyear(it)
528             CALL EXIT(42)
529         end if
530         it = int(it) + 1
531     end do ! Timestep loop
532
533     if(debug > 0 ) then
534         print *, "-----------------"
535     endif
536
537     ! NETCDF: Close file
538     if (write_netcdf) then
539         call closeNCDFFile(filehandle_netcdf)
540     endif
541
542     if(debug > 0 ) then
543         print *, '========================'
544         print *, "Successfully terminated!"
545         print *, "We are at the end"
546         print *, "-----------------"
```

```
547           print *, "Some statistics:"
548           call ETIME(execution_time, runtime)
549           print *, "Runtime [mm:ss]: ", int(runtime/60), ':',
                  ↪ mod(int(runtime),60)
550           print *, "Seconds per 1000 years: ", (runtime/maxyears * 1000)
551           print *, 'yr/hour: ', (maxyears/runtime * 60*60)
552           print *, "-----------------"
553           if (store_input_netcdf .or. write_netcdf) then
554               print *, 'The output is stored in the directory: ',
                      ↪ TRIM(adjustl(output_directory))
555           else
556               print *, 'No output was stored from this run!'
557           endif
558       endif
559
560   end program
```

io_read.f90

The file `io_read.f90` is responsible for the functionality concerning reading from the filesystem.

```
! Module which is responsible for reading the values from the disk.
!
! Author:      neff@climate.unibe.ch
! Date:        2014/5
! Revision:    1.0b
! SVN Info:    $Id: io_read.f90 189 2014-06-01 15:21:53Z basil $


module io_read

include 'netcdf.inc'

CONTAINS

    !---------------------------------------------------------------
    ! NetCDF Stuff
    !---------------------------------------------------------------


    function read_variable(filename, NLONS, NLATS, variable)
        ! Reads the variable (8bit real) from a 2D (long x lat,
           ↪ without time) netcdf file.
        !
        ! With inspirations from here:
        !
           ↪ http://www.unidata.ucar.edu/software/netcdf/examples/programs/simple_x
        !
           ↪ http://www.unidata.ucar.edu/software/netcdf/examples/programs/sfc_pres

        ! Reads
        use netcdf
        use variables
        implicit none

        character(len=*), intent(in) :: filename
        ! input
        integer, intent(in) :: NLATS
        integer, intent(in) :: NLONS
        character(*), intent(in) :: variable

        ! output
        real(kind=8) :: read_variable(NLONS, NLATS)

        ! Return value, to check if everything was ok
        integer :: retval

        ! This will be the netCDF ID for the file and data variable.
        integer :: ncid, varid

        ! Open the file, Read only
```

```
48          ! Trim file path:
               ↪ http://stackoverflow.com/questions/15093712/trimming-string-for-dire
49          retval = nf_open(TRIM(adjustl(filename)), nf_NOWRITE, ncid)
50          if (retval .ne. nf_noerr) call handle_err(retval)
51          if(debug > 4 ) then
52              print *, "io_read.f90: NetCDF File opened: ", filename
53          end if
54
55
56          ! Get the varid of the data variable, based on its name.
57          !retval = nf_inq_varid(ncid, 'LANDMASK', varid)
58          retval = nf_inq_varid(ncid, variable, varid)
59          if (retval .ne. nf_noerr) call handle_err(retval)
60          if(debug > 4 ) then
61              print *, "io_read.f90: NetCDF varid for variable
                   ↪ received: ", variable
62          end if
63
64          ! Read the data as Array, should also work
65          !retval = nf_GET_VARA_DOUBLE(NCID, varid, START, END,
               ↪ landmask_netcdf)
66          !if (retval .ne. nf_noerr) call handle_err(retval)
67          !read_landmask = real(landmask_netcdf, kind=4)
68          ! read the data
69          retval = nf_get_var_double(ncid, varid, read_variable)
70          if (retval .ne. nf_noerr) call handle_err(retval)
71          if(debug > 4 ) then
72              print *, "io_read.f90: NetCDF read the data from the file"
73          end if
74
75          ! Close the file, freeing all resources.
76          retval = nf_close(ncid)
77          if (retval .ne. nf_noerr) call handle_err(retval)
78          if(debug > 4 ) then
79              print *, "io_read.f90: NetCDF file closed"
80          end if
81
82          ! If we got this far, everything worked as expected. Yipee!
83          if(debug > 0) then
84              print *,"*** Successful reading the
                   ↪ ",TRIM(adjustl(variable))," from NetCDF file: ",
                   ↪ TRIM(adjustl(filename)), ""
85          end if
86          return
87      end function read_variable
88
89      function read_watermask(elevation_landmask, NLONS, NLATS,
           ↪ sea_level, elevation_bedrock, ice_thickness)
90          ! Returns a watermask (array, with integers).
91          ! elevation_landmask is the elevation of the bedrock. If this
               ↪ elevation is below the sea level, the grid point is
               ↪ marked as water.
92          ! But if there is ice on the grid point which is heavier than
               ↪ the water column
93          ! (calculated from the difference of the sea level and
               ↪ elevation_bedrock), the grid point will stay as it was.
94          !
```

```
95          ! 1 = water, 0 = land
96
97          !use variables
98
99          !character(len=*), intent(in) :: filename
100         ! input
101         integer, intent(in) :: NLONS
102         integer, intent(in) :: NLATS
103         real(kind=8), intent(in) :: sea_level
104         real(kind=8) :: ice_thickness(NLONS, NLATS)
105         ! output
106         integer :: read_watermask(NLONS, NLATS)
107
108         ! Landmask, to get the position of the water
109         real(kind=8), intent(in) :: elevation_landmask(NLONS, NLATS)
110         real(kind=8), intent(in) :: elevation_bedrock(NLONS, NLATS)
111         ! Calculate the ice column in water equivalent
112         real(kind=8), parameter :: rho_ice = 0.910 ! Eismint, Table 1
113
114         ! Loop indexes
115         integer :: lat, lon
116
117         do lat=1,nlats,1
118             ! then loop over y-axes
119             do lon=1,nlons,1
120                 if (elevation_landmask(lon, lat) .lt. sea_level) then
121                     ! Check if there is ice on the grid point, which
                            ↪ is able to displace the water. Otherwise,
                            ↪ leave it as it is.
122                     if((sea_level - elevation_bedrock(lon, lat)) .gt.
                            ↪ ice_thickness(lon, lat)*rho_ice) then
123                         read_watermask(lon, lat) = 1
124                         ! Otherwise, leave it as it is
125                     end if
126                 elseif(elevation_landmask(lon, lat) .le.
                        ↪ abs(sea_level)) then
127                     ! In case that the sea level incereases during
                            ↪ the run,
128                     ! that not every point (only below 200m) need to
                            ↪ be calculated if there is ice or not.
129                     read_watermask(lon, lat) = 0
130                 endif
131             end do
132         end do
133
134         ! At the end, remove the lakes from the watermask
135         call clean_watermask(read_watermask, NLONS, NLATS)
136
137     end function read_watermask
138
139     subroutine clean_watermask(watermask, NLONS, NLATS)
140         ! Removes the great lakes from the watermask
141         ! Removes isolated water points sourounded by the land.
142         integer, intent(in) :: NLONS
143         integer, intent(in) :: NLATS
144         integer, intent(inout) :: watermask(NLONS, NLATS)
145
```

```
146         ! Remove the lakes in North America
147         if (NLONS .eq. 625) then
148             watermask(60:300, 120:176) = 0
149         else
150             watermask(40:150, 60:88) = 0
151             ! Over whole america
152             watermask(50:120, 1:80) = 0
153             watermask(120:130, 42:50) = 0
154         end if
155
156         ! Remove lakes with the size of one grid box
157         ! 0 = ice
158         ! 1 = water
159         ! 2 = no ice
160         do ix=2,NLONS-1,1
161             do iy=2,NLATS-1,1
162                 ! In X Direction: 2 water boxes, 1 ice boxes
163                 if ( (watermask(ix, iy) .eq. 1) .and.
                        ↪ (watermask(ix-1, iy) .ne. 1) .and.
                        ↪ (watermask(ix+1, iy) .ne. 1) .and.
                        ↪ (watermask(ix, iy-1) .ne. 1) .and.
                        ↪ (watermask(ix, iy+1) .ne. 1) ) then
164                      watermask(ix:ix, iy:iy) = 0
165                 endif
166             end do
167         end do
168
169
170     end subroutine clean_watermask
171
172
173     function read_climate(filename, NLONS, NLATS, variable)
174         ! Reads the temperature or precipitation data (8bit real) for
               ↪ 365 days out of the given NetCDF File.
175         ! variable: prec/temp, units: m/yr and kelvin
176         ! Returns 3D Array: Lon, Lat, Day
177
178         ! Reads
179         use netcdf
180         use variables
181
182         ! input
183         character(len=*), intent(in) :: filename
184         character(len=*), intent(in) :: variable
185         integer, intent(in) :: NLATS
186         integer, intent(in) :: NLONS
187
188         ! Zeitschritte
189         integer, parameter :: days = 365
190
191         ! output
192         ! TODO: Zeitschritt einbinden
193         real(kind=8) :: read_climate(NLONS, NLATS, days)
194
195         ! Return value, to check if everything was ok
196         integer :: retval
197
```

```
198         ! This will be the netCDF ID for the file and data variable.
199         integer :: ncid, varid
200
201         ! Open the file, Read only
202         ! Trim file path:
               ↪ http://stackoverflow.com/questions/15093712/trimming-string-for-direct
203         retval = nf_open(TRIM(adjustl(filename)), nf_NOWRITE, ncid)
204
205         ! Read Data
206         ! Get the varid of the data variable, based on its name.
207         retval = nf_inq_varid(ncid, TRIM(adjustl(variable)), varid)
208         if (retval .ne. nf_noerr) call handle_err(retval)
209
210         ! read the data
211         retval = nf_get_var_double(ncid, varid, read_climate)
212         if (retval .ne. nf_noerr) call handle_err(retval)
213
214         ! Close the file, freeing all resources.
215         retval = nf_close(ncid)
216         if (retval .ne. nf_noerr) call handle_err(retval)
217
218         if(debug > 0) then
219             ! If we got this far, everything worked as expected.
                   ↪ Yipee!
220             print *,"*** Read ", TRIM(adjustl(variable))," from
                   ↪ NetCDF file: ", TRIM(adjustl(filename)), ""
221         end if
222
223         return
224
225     end function read_climate
226
227     subroutine handle_err(errcode)
228         integer errcode
229
230         print *, 'Error: ', nf_strerror(errcode)
231         stop 4
232     end subroutine handle_err
233
234 end module io_read
```

io_write.f90

The file `io_write.f90` is responsible for the functionality concerning writing to the filesystem.

```
! Module which is responsible for writing the values to the disk.
!
! Author:      neff@climate.unibe.ch
! Date:        2014/5
! Revision:    1.0b
! SVN Info:    $Id: io_write.f90 189 2014-06-01 15:21:53Z basil $


module io_write

include 'netcdf.inc'

CONTAINS

!-----------------------------------------------------------------
! Output Directory Stuff
!-----------------------------------------------------------------

subroutine init_output_directory(directory, identifier)
    ! Creates a supdirectory with a timestamp and the identifier of
       ↪ the experiment
    ! Copies the variable and the initial NetCDF files into it.
    ! Sets the new output directory to the variable directory.

    ! Use the variables
    use variables

    ! input
    character(len=*), intent(inout) :: directory
    character(len=*), intent(in) :: identifier
    integer(kind=4)  stime, tarray(9), time

    integer(kind=4)  :: result
    character(len=13) :: timestamp
    character(len=256) :: command
    character(len=256) :: current_directory

    stime = time()
    call ltime(stime, tarray)
    ! YYYYMMDDHHMM :
       ↪ http://docs.oracle.com/cd/E19957-01/805-4942/6j4m3r90k/index.html
    write (timestamp, "(I4,I0.2,I0.2,A1,I0.2, I0.2)") (tarray(6) +
       ↪ 1900), tarray(5)+1, tarray(4), '_',(tarray(3)), tarray(2)
    ! year
    directory = TRIM(adjustl(directory)) // timestamp // '_'
       ↪ //TRIM(adjustl(identifier)) // '/'

    ! create directory
    command = 'mkdir ' // directory
    result = system(command)

```

```
48      if(debug > 0 ) then
49          print *, "*** Output directory created: ",
                ↪ TRIM(adjustl(directory))
50      end if
51
52      ! copy variables
53      CALL getcwd(current_directory)
54      if (index(current_directory, 'home') .gt. 0) then
55          command = 'cp -p ' // TRIM(adjustl(current_directory)) //
                ↪ '/variables.f90 ' // TRIM(adjustl(directory))
56      else
57          command = 'cp -p ' // TRIM(adjustl(current_directory)) //
                ↪ '/../variables.f90 ' // TRIM(adjustl(directory))
58      end if
59      if(debug > 0 ) then
60          print *, "*** Variables copied: ", TRIM(adjustl(command))
61      end if
62      result = system(command)
63  end subroutine init_output_directory
64
65  subroutine copy_to_output_directory(filepath, directory)
66
67      ! Copies the given file to the output directory
68      character(len=*), intent(in) :: filepath
69      character(len=*), intent(in) :: directory
70
71      integer(kind=4)   :: result
72      character(len=256) :: command
73
74      command = 'cp -p ' // TRIM(adjustl(filepath)) // ' ' //
            ↪ TRIM(adjustl(directory))
75      result = system(command)
76
77  end subroutine copy_to_output_directory
78
79
80  !---------------------------------------------------------------
81  ! NetCDF Stuff
82  !---------------------------------------------------------------
83
84  subroutine init_netcdf_file(filename, ncid, nlats, nlons,
        ↪ lat_distance, long_distance, height_varid, bed_varid,
        ↪ acc_varid, abl_varid, diffusivity_varid, discharge_x_varid,
        ↪ discharge_y_varid, terrain_mask_varid)
85      ! Init ncdf file
86      ! param filename: path and filename to the file
87      ! param ncid: filehandle for the ncid file, will be overwritten
88      ! param nlats: Number of latititudes (integer)
89      ! param nlons: Number of longitued (integer)
90      ! param lat_distance: Length in m between the points of latitude
91      ! param long_distance: Length in m between the points of longitude
92      !
93      ! Output
94      ! param height_varid: Variable ID if the height (needed to write
            ↪ valued to the netCDF)
95      ! param bed_varid: Variable ID of the bedrock values
96      ! param acc_varid: Variable ID of the accumulation
```

```
97      ! param abl_varid: Variable ID of the ablation
98      ! param diffusivity_varid: Variable ID for the diffusivity
99      ! param discharge_x_varid: Variable ID of the discharge in x
           ↪ direction
100     ! param discharge_y_varid: Variable ID of the discharge in y
           ↪ direction
101     ! param terrain_mask_varid: Mask for calculations (0 = ice, 1 =
           ↪ water, 2 = land without ice)
102     !
103     ! Initialise the variables to write in 2D data with time
           ↪ dependency
104     ! Prepare the following variables to write into the file:
105     ! - Thickness (2D Real Array): varid =
106     ! - Bedrock (2D Real Array)
107     ! - Elevation Line Position (2D Real Array)
108     ! - Mass Balance (in preparation)
109
110
111     use netcdf
112     character(len=*), intent(in) :: filename
113     integer, intent(out) :: ncid
114     ! input
115     integer, intent(in) :: NLATS
116     integer, intent(in) :: NLONS
117     integer, intent(in) :: lat_distance
118     integer, intent(in) :: long_distance
119     ! output
120     integer, intent(out) :: height_varid        ! Variable for the
           ↪ netCDF Height, parameter
121     integer, intent(out) :: bed_varid           ! Variable for the
           ↪ netCDF Bedrock, parameter
122     integer, intent(out) :: acc_varid           ! Variable for the
           ↪ netCDF accumulation, parameter
123     integer, intent(out) :: abl_varid           ! Variable for the
           ↪ netCDF ablation, parameter
124     integer, intent(out) :: diffusivity_varid   ! Variable for the
           ↪ netCDF Diffusivity, parameter
125     integer, intent(out) :: discharge_x_varid   ! Variable for the
           ↪ netCDF discharge in x direction, parameter
126     integer, intent(out) :: discharge_y_varid   ! Variable for the
           ↪ netCDF discharge in y direction, parameter
127     integer, intent(out) :: terrain_mask_varid! Variable for the
           ↪ netCDF terrain_mask, parameter
128
129
130     ! return value
131     integer :: retval
132
133     ! Copied from Example file!
134     !--------------------------
135     ! We are writing 2D data with time, We will need 3 netCDF
           ↪ dimensions. (Lats, Long, Time)
136     integer, parameter :: NDIMS = 3
137
138     character*(*) :: LAT_NAME, LON_NAME, REC_NAME
139     parameter (LAT_NAME='latitude', LON_NAME='longitude')
140     parameter (REC_NAME = 'time')
```

```
141     integer lon_dimid, lat_dimid, rec_dimid
142
143     ! The start and count arrays will tell the netCDF library where to
144     ! write our data.
145     !integer start(NDIMS), count(NDIMS)
146
147     ! In addition to the latitude and longitude dimensions, we will
            ↪ also
148     ! create latitude and longitude netCDF variables which will hold
            ↪ the
149     ! actual latitudes and longitudes. Since they hold data about the
150     ! coordinate system, the netCDF term for these is: "coordinate
151     ! variables."
152     real :: lats(NLATS)
153     real :: lons(NLONS)
154     integer lat_varid, lon_varid, rec_varid
155     real START_LAT, START_LON
156     parameter (START_LAT = 0, START_LON = 0)
157 !
158 !     ! Variables
159     character*(*) HEIGHT_NAME, BED_NAME, ACC_NAME, ABL_NAME,
            ↪ DIFFUSIVITY_NAME
160     character*(*) DISCHARGE_X_NAME, DISCHARGE_Y_NAME,
            ↪ ASSIGNMENT_MASK_NAME
161     parameter (HEIGHT_NAME='height')
162     parameter (BED_NAME='bedrock')
163     parameter (ACC_NAME='accumulation')
164     parameter (ABL_NAME='ablation')
165     parameter (DIFFUSIVITY_NAME='diffusivity')
166     parameter (DISCHARGE_X_NAME = 'discharge_x')
167     parameter (DISCHARGE_Y_NAME = 'discharge_y')
168     parameter (ASSIGNMENT_MASK_NAME = 'terrain_mask')
169     integer dimids(NDIMS)
170 !
171 !     ! It's good practice for each variable to carry a "units"
        ↪ attribute.
172     character*(*) :: UNITS
173     parameter (UNITS = 'units')
174     character*(*) HEIGHT_UNITS, BED_UNITS, ACC_UNITS, ABL_UNITS,
            ↪ LAT_UNITS, LON_UNITS, REC_UNITS, DIFFUSIVITY_UNITS,
            ↪ DISCHARGE_UNITS
175     parameter (HEIGHT_UNITS = 'm', BED_UNITS = 'm', ACC_UNITS =
            ↪ 'm/yr', ABL_UNITS = 'm/yr', DIFFUSIVITY_UNITS = 'm2/yr',
            ↪ DISCHARGE_UNITS = 'm2/yr')
176     ! Units of Dimensions
177     parameter (LAT_UNITS = 'm')
178     parameter (LON_UNITS = 'm')
179     parameter (REC_UNITS = '500years')
180
181     ! Distance of the lat/long points
182     do lat = 1, NLATS
183        lats(lat) = START_LAT + (lat - 1) * lat_distance
184     end do
185     do lon = 1, NLONS
186        lons(lon) = START_LON + (lon - 1) * long_distance
187     end do
188
```

```
189       !print *, '*** Init NetCDF file: ', TRIM(adjustl(filename))
190
191       ! Create the file.
192       retval = nf_create(filename, nf_clobber, ncid)
193       if (retval .ne. nf_noerr) call handle_err(retval)
194   !
195   !
196   !     ! Define dimensions
197       retval = nf_def_dim(ncid, LAT_NAME, NLATS, lat_dimid)
198       if (retval .ne. nf_noerr) call handle_err(retval)
199       retval = nf_def_dim(ncid, LON_NAME, NLONS, lon_dimid)
200       if (retval .ne. nf_noerr) call handle_err(retval)
201       ! TODO Time - this should be OK
202       retval = nf_def_dim(ncid, REC_NAME, NF_UNLIMITED, rec_dimid)
203       if (retval .ne. nf_noerr) call handle_err(retval)
204
205       ! Define the coordinate variables. They will hold the coordinate
206       ! information, that is, the latitudes and longitudes. A varid is
207       ! returned for each.
208       !retval = nf_def_var(ncid, LAT_NAME, NF_REAL, 1, lat_dimid,
              ↪ lat_varid)
209       retval = nf_def_var(ncid, LAT_NAME, NF_DOUBLE, 1, lat_dimid,
              ↪ lat_varid)
210       if (retval .ne. nf_noerr) call handle_err(retval)
211       !retval = nf_def_var(ncid, LON_NAME, NF_REAL, 1, lon_dimid,
              ↪ lon_varid)
212       retval = nf_def_var(ncid, LON_NAME, NF_DOUBLE, 1, lon_dimid,
              ↪ lon_varid)
213       if (retval .ne. nf_noerr) call handle_err(retval)
214       ! TODO Time
215       !retval = nf_def_var(ncid, REC_NAME, NF_REAL, 1, rec_dimid,
              ↪ rec_varid)
216       retval = nf_def_var(ncid, REC_NAME, NF_DOUBLE, 1, rec_dimid,
              ↪ rec_varid)
217       if (retval .ne. nf_noerr) call handle_err(retval)
218
219       ! Assign units attributes to coordinate var data. This attaches a
220       ! text attribute to each of the coordinate variables, containing
              ↪ the
221       ! units.
222       retval = nf_put_att_text(ncid, lat_varid, UNITS, len(LAT_UNITS),
              ↪ LAT_UNITS)
223       if (retval .ne. nf_noerr) call handle_err(retval)
224       retval = nf_put_att_text(ncid, lon_varid, UNITS, len(LON_UNITS),
              ↪ LON_UNITS)
225       if (retval .ne. nf_noerr) call handle_err(retval)
226       ! TODO Time
227       retval = nf_put_att_text(ncid, rec_varid, UNITS, len(REC_UNITS),
              ↪ REC_UNITS)
228       if (retval .ne. nf_noerr) call handle_err(retval)
229
230       ! Define the netCDF variables. The dimids array is used to pass
              ↪ the
231       ! dimids of the dimensions of the netCDF variables.
232       dimids(1) = lon_dimid
233       dimids(2) = lat_dimid
234       dimids(3) = rec_dimid
```

```
235
236     ! Define the netCDF variables for the "real" values
237     !retval = nf_def_var(ncid, HEIGHT_NAME, NF_REAL, NDIMS, dimids,
            ↪ height_varid)
238     retval = nf_def_var(ncid, HEIGHT_NAME, NF_DOUBLE, NDIMS, dimids,
            ↪ height_varid)
239     if (retval .ne. nf_noerr) call handle_err(retval)
240     !retval = nf_def_var(ncid, BED_NAME, NF_REAL, NDIMS, dimids,
            ↪ bed_varid)
241     retval = nf_def_var(ncid, BED_NAME, NF_DOUBLE, NDIMS, dimids,
            ↪ bed_varid)
242     if (retval .ne. nf_noerr) call handle_err(retval)
243     !retval = nf_def_var(ncid, ACC_NAME, NF_REAL, NDIMS, dimids,
            ↪ ACC_varid)
244     retval = nf_def_var(ncid, ACC_NAME, NF_DOUBLE, NDIMS, dimids,
            ↪ acc_varid)
245     if (retval .ne. nf_noerr) call handle_err(retval)
246     !retval = nf_def_var(ncid, ABL_NAME, NF_REAL, NDIMS, dimids,
            ↪ ACC_varid)
247     retval = nf_def_var(ncid, ABL_NAME, NF_DOUBLE, NDIMS, dimids,
            ↪ abl_varid)
248     if (retval .ne. nf_noerr) call handle_err(retval)
249     !retval = nf_def_var(ncid, DIFFUSIVITY_NAME, NF_REAL, NDIMS,
            ↪ dimids, diffusivity_varid)
250     retval = nf_def_var(ncid, DIFFUSIVITY_NAME, NF_DOUBLE, NDIMS,
            ↪ dimids, diffusivity_varid)
251     if (retval .ne. nf_noerr) call handle_err(retval)
252     !retval = nf_def_var(ncid, DISCHARGE_X_NAME, NF_REAL, NDIMS,
            ↪ dimids, discharge_x_varid)
253     retval = nf_def_var(ncid, DISCHARGE_X_NAME, NF_DOUBLE, NDIMS,
            ↪ dimids, discharge_x_varid)
254     if (retval .ne. nf_noerr) call handle_err(retval)
255     !retval = nf_def_var(ncid, DISCHARGE_Y_NAME, NF_REAL, NDIMS,
            ↪ dimids, discharge_y_varid)
256     retval = nf_def_var(ncid, DISCHARGE_Y_NAME, NF_DOUBLE, NDIMS,
            ↪ dimids, discharge_y_varid)
257     if (retval .ne. nf_noerr) call handle_err(retval)
258     retval = nf_def_var(ncid, ASSIGNMENT_MASK_NAME, NF_INT, NDIMS,
            ↪ dimids, terrain_mask_varid)
259     if (retval .ne. nf_noerr) call handle_err(retval)
260
261
262     ! Assign units attributes to the pressure and temperature netCDF
263     ! variables.
264     retval = nf_put_att_text(ncid, height_varid, UNITS,
            ↪ len(HEIGHT_UNITS), HEIGHT_UNITS)
265     if (retval .ne. nf_noerr) call handle_err(retval)
266     retval = nf_put_att_text(ncid, bed_varid, UNITS, len(BED_UNITS),
            ↪ BED_UNITS)
267     if (retval .ne. nf_noerr) call handle_err(retval)
268     retval = nf_put_att_text(ncid, acc_varid, UNITS, len(ACC_UNITS),
            ↪ acc_UNITS)
269     if (retval .ne. nf_noerr) call handle_err(retval)
270     retval = nf_put_att_text(ncid, abl_varid, UNITS, len(ABL_UNITS),
            ↪ abl_UNITS)
271     if (retval .ne. nf_noerr) call handle_err(retval)
```

```
272     retval = nf_put_att_text(ncid, diffusivity_varid, UNITS,
            ↪ len(DIFFUSIVITY_UNITS), DIFFUSIVITY_UNITS)
273     if (retval .ne. nf_noerr) call handle_err(retval)
274     retval = nf_put_att_text(ncid, discharge_x_varid, UNITS,
            ↪ len(DISCHARGE_UNITS), DISCHARGE_UNITS)
275     if (retval .ne. nf_noerr) call handle_err(retval)
276     retval = nf_put_att_text(ncid, discharge_y_varid, UNITS,
            ↪ len(DISCHARGE_UNITS), DISCHARGE_UNITS)
277     if (retval .ne. nf_noerr) call handle_err(retval)
278
279     ! End define mode.
280     retval = nf_enddef(ncid)
281     if (retval .ne. nf_noerr) call handle_err(retval)
282
283     ! Write the coordinate variable data. This will put the latitudes
284     ! and longitudes of our data grid into the netCDF file.
285     retval = nf_put_var_real(ncid, lat_varid, lats)
286     if (retval .ne. nf_noerr) call handle_err(retval)
287     retval = nf_put_var_real(ncid, lon_varid, lons)
288     if (retval .ne. nf_noerr) call handle_err(retval)
289     print *,'*** Output netCDF file defined: ',
            ↪ TRIM(adjustl(filename))
290
291 end subroutine init_netcdf_file
292
293
294 subroutine closeNCDFFile(ncid)
295     use netcdf
296     integer, intent(in) :: ncid
297
298     integer :: retval
299
300     ! Close the file.
301     retval = nf_close(ncid)
302     if (retval .ne. nf_noerr) call handle_err(retval)
303
304     ! If we got this far, everything worked as expected. Yipee!
305     print *,'*** netCDF file written!'
306
307 end subroutine closeNCDFFile
308
309
310
311 subroutine writeNCDFGridValues(ncid, year, varid, values, nlats,
    ↪ nlons)
312     ! Writes the Values (Real(kind=8)) into the netCDF File
313     ! Param ncid: File handle of the netCDF File
314     ! Param year: The year of the symmulation (integer)
315     ! Param varid: Variable ID (a, bedrock, ...) the values belong to
316     ! Param values: Array (lon, lat) with values as REAL(kind=8)
317     ! param nlats: Number of latititudes (integer)
318     ! param nlons: Number of longitued (integer)
319     !
320     ! Flushs the data to the disk afterwards
321     use netcdf
322
323     ! Parameters
```

```
324     integer, intent(in) :: ncid
325     integer, intent(in) :: year
326     integer, intent(in) :: varid
327     integer, intent(in) :: nlats
328     integer, intent(in) :: nlons
329     real(kind=8), dimension(nlons,nlats), intent(in) :: values
330
331     ! Error handling.
332     integer :: retval
333
334     ! We are writing 2D data with time, We will need 3 netCDF
            ↪ dimensions. (Lats, Long, Time)
335     integer, parameter :: NDIMS = 3
336
337     ! The start and count arrays will tell the netCDF library where to
338     ! write our data.
339     integer start(NDIMS), count(NDIMS)
340
341
342     ! These settings tell netcdf to write one timestep of data. (The
343     ! setting of start(3) inside the loop below tells netCDF which
344     ! timestep to write.)
345     count(1) = NLONS
346     count(2) = NLATS
347     count(3) = 1
348     start(1) = 1
349     start(2) = 1
350     start(3) = year
351
352     ! Write the pretend data. This will write the data.
353     ! The arrays only hold one timestep worth of data.
354     !retval = nf_put_vara_real(ncid, varid, start, count, values)
355     retval = nf_put_vara_double(ncid, varid, start, count, values)
356     if (retval .ne. nf_noerr) call handle_err(retval)
357
358     ! Flush data to the disk
359     retval = NF_SYNC(ncid)
360     if (retval .ne. nf_noerr) call handle_err(retval)
361
362 end subroutine writeNCDFGridValues
363
364
365 subroutine writeNCDFGridIntegerValues(ncid, year, varid, values,
        ↪ nlats, nlons)
366     ! Writes the Values (Real(kind=8)) into the netCDF File
367     ! Param ncid: File handle of the netCDF File
368     ! Param year: The year of the symmulation (integer)
369     ! Param varid: Variable ID (a, bedrock, ...) the values belong to
370     ! Param values: Array (lon, lat) with values as Integer
371     ! param nlats: Number of latititudes (integer)
372     ! param nlons: Number of longitued (integer)
373     !
374     ! Flushs the data to the disk afterwards
375     use netcdf
376
377     ! Parameters
378     integer, intent(in) :: ncid
```

```
379     integer, intent(in) :: year
380     integer, intent(in) :: varid
381     integer, intent(in) :: nlats
382     integer, intent(in) :: nlons
383     integer, dimension(nlons,nlats), intent(in) :: values
384
385     ! Error handling.
386     integer :: retval
387
388     ! We are writing 2D data with time, We will need 3 netCDF
        ↪ dimensions. (Lats, Long, Time)
389     integer, parameter :: NDIMS = 3
390
391     ! The start and count arrays will tell the netCDF library where to
392     ! write our data.
393     integer start(NDIMS), count(NDIMS)
394
395
396     ! These settings tell netcdf to write one timestep of data. (The
397     ! setting of start(3) inside the loop below tells netCDF which
398     ! timestep to write.)
399     count(1) = NLONS
400     count(2) = NLATS
401     count(3) = 1
402     start(1) = 1
403     start(2) = 1
404     start(3) = year
405
406     ! Write the pretend data. This will write the data.
407     ! The arrays only hold one timestep worth of data.
408     !retval = nf_put_vara_real(ncid, varid, start, count, values)
409     retval = nf_put_vara_int(ncid, varid, start, count, values)
410     if (retval .ne. nf_noerr) call handle_err(retval)
411
412     ! Flush data to the disk
413     retval = NF_SYNC(ncid)
414     if (retval .ne. nf_noerr) call handle_err(retval)
415
416 end subroutine writeNCDFGridIntegerValues
417
418
419 subroutine handle_err(errcode)
420       integer errcode
421
422       print *, 'Error: ', nf_strerror(errcode)
423       stop 2
424 end subroutine handle_err
425
426
427 end module io_write
```

smb.f90

All functions concerning the surface mass balance can be found in the following file `smb.f90`.

```
! Own module to get the values from the external forcing
! Author:        neff@climate.unibe.ch
! Date:          2014/05
! Revision:      1.0b
! SVN Info:      $Id: smb.f90 189 2014-06-01 15:21:53Z basil $

module smb

CONTAINS


    function temperature_dependent_accumulation(nx, ny,
       ↪ temperature_array, precipitation_array, precipitation_unit,
       ↪ daily_temperature_threshold)
        ! Used to calculate the accumulation over a specific domain
           ↪ given by the precipitation and temperature over the
           ↪ year.

        ! Static parameters
        integer, parameter :: days = 365
        real(kind=8), parameter :: kelvin = 273.15
        !real(kind=8), parameter :: daily_temperature_threshold = 0 !
           ↪ in Celsius

        ! input
        real(kind=8), intent(in) :: daily_temperature_threshold
        integer, intent(in) :: NX
        integer, intent(in) :: NY
        ! Temperature at the level of the ice elevation
        real(kind=8), intent(in) :: temperature_array(nx, ny, days)
        ! Precipitation
        real(kind=8), intent(in) :: precipitation_array(nx, ny, days)
        integer, intent(in) :: precipitation_unit
        ! Output
        real(kind=8) :: temperature_dependent_accumulation(nx, ny)


        ! accumulation out of the precipitation
        ! Precipication in m
        temperature_dependent_accumulation = 0

        do id=1,days,1
            !
               ↪ http://www.stanford.edu/class/me200c/tutorial_90/07_arrays.html
            where(temperature_array(:,:,id) .lt.
               ↪ (daily_temperature_threshold + kelvin))
                ! Accumulation: All elements in the array, where the
                   ↪ temperature is below 0 degree celsius
                temperature_dependent_accumulation =
                   ↪ temperature_dependent_accumulation +
                   ↪ precipitation_array(:,:, id)
```

```
42              end where
43          end do
44
45          ! Accumulation: m/yr to m/s
46          if(precipitation_unit == 1) then
47              temperature_dependent_accumulation =
                    ↪ temperature_dependent_accumulation / (days * 24 *
                    ↪ 60 * 60)
48          endif
49          ! m/s divide through the days of the integrated year
50          if(precipitation_unit == 2) then
51              temperature_dependent_accumulation =
                    ↪ temperature_dependent_accumulation / days
52          endif
53
54      end function temperature_dependent_accumulation
55
56      function temperature_dependent_ablation(nx, ny,
            ↪ temperature_array, beta)
57          ! Ablation calculated on the princible of the positive degree
                ↪ days (PDD).
58          !
59          ! nx and ny: Size of the 2d array
60          ! temperature_array: (nx, ny, 365): Temperature of the grid
                ↪ point all over the year at the level of the ice
                ↪ elevation in kelvin.
61
62          ! Static parameters
63          integer, parameter :: days = 365
64          real(kind=8), parameter :: kelvin = 273.15
65          real(kind=8), parameter :: daily_temperature_threshold = 0 !
                ↪ in Celsius
66          ! http://www.igsoc.org/journal/59/218/j13J081.pdf
67          ! 3 mm * C^-1 * d^-1 = 3.47e-8 m * C^-1 * s^-1 for snow
68          ! 8 mm * C^-1 * d^-1 = 9.26e-8 m * C^-1 * s^-1 for ice
69          ! 3 - 8 mm * C^-1 * d^-1 = 3 / (1000 * 24 * 60 * 60) = 3.5e-8
                ↪ m * C^-1 * s^-1
70          real(kind=8), intent(in) :: beta ! = 6e-8/days
71
72          ! input
73          integer, intent(in) :: NX
74          integer, intent(in) :: NY
75          ! Temperature at the level of the ice elevation
76          real(kind=8), intent(in) :: temperature_array(nx, ny, days)
77          ! Output
78          real(kind=8) :: temperature_dependent_ablation(nx, ny)
79          ! Days with a temperature over 0 degrees
80          real(kind=8) :: positive_degree_days(nx, ny)
81
82          ! accumulation out of the precipitation
83          ! Calculate positive degree days
84          positive_degree_days = 0
85
86          do id=1,days,1
87              !
                    ↪ http://www.stanford.edu/class/me200c/tutorial_90/07_arrays.html
```

```
88              where(temperature_array(:,:,id) .ge. kelvin +
                    ↪ daily_temperature_threshold)
89                  ! Ablation (PDD): All elements, where the temperature
                        ↪ is above the temperature threshhold degree
                        ↪ Celsius
90                  positive_degree_days = positive_degree_days +
                        ↪ (temperature_array(:,:,
                        ↪ id)-kelvin-daily_temperature_threshold)
91              end where
92          end do
93
94          ! Calculate accumulation/ablation out of it
95          temperature_dependent_ablation = (beta * positive_degree_days
                ↪ / days)
96          return
97      end function temperature_dependent_ablation
98
99
100     subroutine ignore_himalaya_accumulation(accumulation, nx, ny)
101         ! Sets the accumultion in the himalaya region to 0.
102         ! This can also be used for the ablation
103
104         integer, intent(in) :: NX
105         integer, intent(in) :: NY
106         real(kind=8), intent(inout) :: accumulation(nx, ny)
107
108         if (nx .eq. 625) then
109             accumulation(440:610, 480:620) = 0
110         else
111             accumulation(220:305, 240:310) = 0
112         end if
113
114     end subroutine ignore_himalaya_accumulation
115
116 end module smb
```

common.f90

The file `common.f90` contains all stuff which is used at different locations of the application. At the moment only the calculations concerning the sea level is included there.

```
1  ! Module with functions and subroutines, which are not assigned at a
       ↪ specifig use case.
2  !
3  ! Author:      neff@climate.unibe.ch
4  ! Date:        2014/5
5  ! Revision:    1.0b
6  ! SVN Info:    $Id: common.f90 191 2014-06-02 16:03:29Z basil $
7
8
9  module common
10
11 CONTAINS
12
13     function get_sea_level(ice_thickness, nx, ny, dx, dy, ocean_area)
14         ! calculates the sea level,
15         ! with the assumption that no ice is equal to a sea level of
               ↪ 0, ice leads to a negative sea level
16         ! use the parameter sea_level_offset in variables.f90 to
               ↪ adjust to sea level without ice
17
18         ! input
19         integer, intent(in) :: NX
20         integer, intent(in) :: NY
21         integer, intent(in) :: DX
22         integer, intent(in) :: DY
23         ! Ice thickness
24         real(kind=8), intent(in) :: ice_thickness(nx, ny)
25         ! Area of the ocean
26         real(kind=8), intent(in) :: ocean_area
27         ! Output
28         real(kind=8) :: get_sea_level
29
30         ! Sea level: Volume/Area = height
31         get_sea_level = -(sum(Ice_thickness) * dx * dy)/ocean_area
32
33         return
34     end function get_sea_level
35
36 end module common
```

Acknowledgement

First of all I would like to thank Andreas Born. This work would not exist in the same way without his very fruitful input, constructive feedback and positive motivation. His assistance was invaluable and saved me from many struggles and mental stress especially in the last weeks.

I also would like to express my gratitude to Thomas Stocker for supervising this thesis, taking the time and provide me with new input and feedback at a time of appointments in every timezone.

Juliette Mignot together with Andreas were responsible for a pleasant working environment.

Many members of the KUP staff donated CPU power of their Linux clients to me to run all my corrected simulations in the last minute again. The simulations would probably still run on my client without their CPU cores: Cevahir, Claudio, Gianna, Juan-José, Martina, Melanie, Michael, René and Sandro.

I would also thank the table soccer crew for the beautiful distraction: Cevahir, Flavio, Nick, Raphael. And all other members of the KUP for the technical and nontechnical discussions during the coffee breaks.

And I would like to see into the future and acknowledge everyone who uses the IceBern2D model and acknowledges me in this work.

Declaration

under Art. 28 Para. 2 RSL 05

Last, first name:	Neff, Basil
Matriculation number:	08-064-016
Programme:	M.Sc. in Climate Sciences
	Bachelor ☐ Master ☑ Dissertation ☐
Thesis title:	IceBern2D: An Efficient Two-Dimensional Ice Sheet Model for Paleoclimate Studies.
Thesis supervisor:	Prof. Dr. Thomas Stocker

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person, except where due acknowledgement has been made in the text. In accordance with academic rules and ethical conduct, I have fully cited and referenced all material and results that are not original to this work. I am well aware of the fact that, on the basis of Article 36 Paragraph 1 Letter o of the University Law of 5 September 1996, the Senate is entitled to deny the title awarded on the basis of this work if proven otherwise.

Bern, 18 June 2014